

MAY 1990

COMPUTER

RECENT
DEVELOPMENTS
IN
**OPERATING
SYSTEMS**



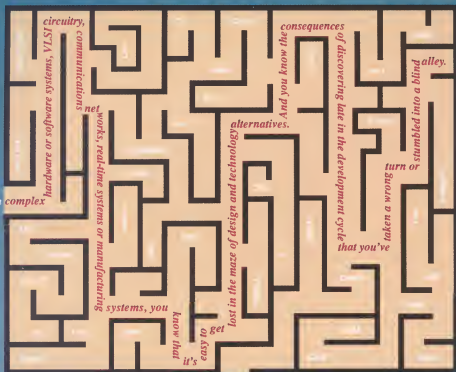
IEEE COMPUTER SOCIETY



THE INSTITUTE OF ELECTRICAL AND
ELECTRONICS ENGINEERS, INC.

If you design complex

**Turn Your
Designing
Maze Into
Amazing
Designs.**



Oops!

SES/workbench™

System-Level Design Simulation and Modeling Software

Now there is an easy to use, elegant tool that guides you through the maze of complex systems design and helps you evaluate design alternatives to discover the consequences of decisions *before* you commit to them. **SES/workbench.**

SES/workbench is the premier multi-level design environment from the established leaders in system simulation, modeling and design evaluation, Scientific and Engineering Software, Inc. It can be used to evaluate design decisions throughout the development cycle, from the earliest conceptual stages, through system specification and design, to functional and performance verification.

SES/workbench advances the state of the art in system-level design and evaluation software.

• Its unique graphical interface, **SES/design**, allows you to create a pictorial representation of a system's structure and to specify its behavior at a high level.

• You can quickly analyze design alternatives and tradeoffs with a few clicks of a mouse button.

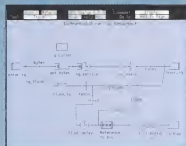
• Representation of the system evolves naturally from a high-level behavioral description to a low-level structural description.

• The graphical representation of the system is translated into **SES/sim**, an object-based superset of C and C++ that offers many extensions developed specifically for simulation modeling.

• Comprehensive statistical reports provide a precise description of the system's behavior, so that you can evaluate system performance.

Free trial licenses are available. Call or write to us for additional information.

SES/workbench.
A tested bed for the imagination.



Using **SES/design** you can easily construct graphical representations of highly complex systems. **SES/design** graphs are automatically translated into **SES/sim**, the powerful simulation language at the heart of **SES/workbench**.



SES/design provides a rich set of graphical modeling primitives that are the basic building blocks for system level design.

Scientific and Engineering Software, Inc.
1301 West 25th Street, Suite #300
Austin, Texas, U.S.A.
Phone: 512/474-4526 Fax: 512/479-6217



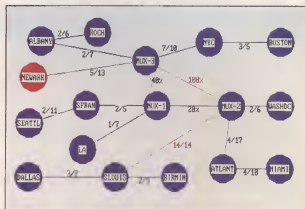
New for network analysts and designers

Free trial and, if you act now, free training

LANNET II.5 Local Area Networks



COMNET II.5 Wide Area Networks



LANNET II.5 uses simulation to predict your LAN performance. You simply describe your LAN and workload.

Animated simulation follows immediately --no programming.

Easy-to-understand results

You get an animated picture of your LAN. System bottlenecks and changing levels of utilization are apparent.

Your reports show LAN statistics such as transfer times, delays, and queues. Client, server, and gateway statistics show queue lengths, waiting times, and messages sent.

Your LAN simulated

You can predict the performance of any LAN. Industry standard protocols such as Ethernet, Token Ring, and Token Bus are built-in. Variations can be modeled.

COMNET II.5 uses simulation to predict your network performance. You simply describe your network, traffic load, and routing algorithms.

Animated simulation follows immediately --no programming.

Easy-to-understand results

You get an animated picture of your network. Routing choices and changing levels of network utilization are apparent.

Your reports show response times, blocking probabilities, call queueing and packet delays, network throughput, circuit group utilization, and circuit group queue statistics.

Your network simulated

You can predict the performance of any communication network--circuit, message, or packet switching. Virtual-circuit or datagram operation can be modeled.

Free 60 Day Trial Offer

The free trial contains everything you need to try LANNET II.5™ or COMNET II.5™ on your PC, Workstation, or Mainframe. Act now for free training--no cost, no obligation.

For immediate information

Call Cliff Baker at (619) 457-9681, Fax (619) 457-1184. In Europe, call Nigel McNamara, in the UK, on (01) 332-0122, Fax (01) 332-0112. In Canada, call (613) 747-7467, Fax (613) 747-2224.

University faculty should call about our special offer for research and teaching.

Rush free trial & training information for:

☐ LANNET II.5

☐ COMNET II.5

Name

Organization

Address

Telephone

Fax

Computer

Operating System

Return to:
CACI Products Co.
3344 N. Torrey Pines Ct.
La Jolla, CA 92037
Call Cliff Baker
at (619) 457-9681.
Fax (619) 457-1184.

In Europe:
CACI Products Div.
Palm Ct., 4 Heron Sq.
Richmond-Upon-Thames
Surrey TW9 1EW, UK
Call Nigel McNamara
on (01) 332-0122.
Fax (01) 332-0112.

In Canada:
CACI Products Co.
1545 Garding Ave.
Ottawa, Ontario
Canada K1Z 8P9
Call (613) 747-7467.
Fax (613) 747-2224.

IEEE COMP

COMPUTER

May 1990

Published by the IEEE Computer Society

Vol. 23, No. 5

ARTICLES

5 Guest Editors' Introduction: Recent Developments in Operating Systems

Joseph Boykin and Susan J. LoVerso

9 Scalable, Secure, and Highly Available Distributed File Access

Mahadev Satyanarayanan

Andrew and Coda are distributed Unix file systems that embody many of the recent advances in solving the problem of data sharing in large, physically dispersed workstation environments.

23 The x-kernel: A Platform for Accessing Internet Resources

Larry Peterson, Norman Hutchinson, Sean O'Malley, and Herman Rao

The x-kernel gives workstation users uniform access to resources across local or wide area networks. The operating system is general, efficient, and easy to program.

35 Scheduling Support for Concurrency and Parallelism in the Mach Operating System

David L. Black

Traditional time-sharing schedulers are inadequate for parallel and concurrent programs, which require new techniques such as processor allocation and handoff scheduling.

44 Amoeba: A Distributed Operating System for the 1990s

Sape J. Mullender, Guido van Rossum, Andrew S. Tanenbaum, Robert van Renesse, and Hans van Staveren

The Amoeba distributed operating system appears to users as a centralized system, but it has the speed, fault tolerance, security safeguards, and flexibility required for the 1990s.

54 Algorithms Implementing Distributed Shared Memory

Michael Stumm and Songnian Zhou

This article compares several algorithms for implementing distributed shared memory. It shows that the performance of these algorithms is sensitive to the memory access behavior of applications.

65 Distributed Hierarchical Control for Parallel Processing

Dror G. Feitelson and Larry Rudolph

A novel design using a hierarchy of controllers effectively controls a multiuser, multiprogrammed parallel system. Such a structure allows dynamic repartitioning according to changing job requirements.

Circulation: *Computer* (ISSN 0018-9162) is published monthly by the IEEE Computer Society, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264; phone (714) 821-8380. IEEE Computer Society Headquarters, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903; IEEE Headquarters, 345 East 47th St., New York, NY 10017. Annual subscription included in society member dues. Nonmember subscription rates: available upon request. Single copy prices: members \$10.00; nonmembers \$20.00. This magazine is also available in microfiche form.

Postmaster: Send undelivered copies and address changes to *Computer*, IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854. Second class postage is paid at New York, New York, and at additional mailing offices.

Copyright and reprint permission: Copyright © 1990 by the Institute of Electrical and Electronics Engineers, Inc. All rights reserved. Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of US copyright law for private use of patrons: (1) those post-1977 articles that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 29 Congress Street, Salem, MA 01970; (2) pre-1978 articles without fee. Instructions are permitted to photocopy isolated articles for noncommercial classroom use without fee. For other copying, reprint, or republication permission, write to Permissions Editor, *Computer*, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264.

Editorial: Unless otherwise stated, bylined articles, as well as product and service descriptions, reflect the author's or firm's opinion. Inclusion in *Computer* does not necessarily constitute endorsement by the IEEE or the Computer Society. All submissions are subject to editing for style, clarity, and space considerations.



DEPARTMENTS

79 Standards

How not to write commercial standards

82 Computer Society News

BoG adopts statement on IEEE reorganization; Award winners announced at Compcon; Gordon Bell Prize winners announced; TC provides forum for practical AI

87 Product Reviews

Graphical word processors; Getting the most for your laser printer dollars

97 New Products

100 IC/Microsystem Announcements

104 Conferences

Compcon Spring 90; NCGA 90; parallel processing; design automation

109 Call for Papers/Calendar

124 Book Reviews

127 New Literature/CS Magazines

128 Open Channel

The laws of statistics



Cover photo: Gabe Palmer © Mug Shots

Cover design: Jay Simpson, Design & Direction

In the next issue

Cache architectures for tightly coupled multiprocessors

Career Opportunities, 118; Computer Society Information, 123; Membership Application, 34; Change-of-Address Form, 126; Advertiser/Product Index, 96; Reader Service Card, 96A

COMPUTER

10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264

Editor-in-Chief

Bruce D. Shriver, University of Southwestern Louisiana

Associate Technical Editor

Jon T. Butler, Naval Postgraduate School

Editorial Board

Dharma P. Agrawal, North Carolina State University

Dennis R. Allison, Stanford University

Fletcher Buckley, GE Aerospace

Scott Davidson, AT&T

Richard Eckhouse, MOCO

Scott E. Fahlman, Carnegie Mellon University

Edmund Galizzi, Eckerd College

John R. Gurd, University of Manchester

Brent T. Hallpern, IBM T.J. Watson Research Center

James H. Haynes, University of California, Santa Cruz

Guy Johnson, Rochester Institute of Technology

Yale Patt, University of Michigan

Guylaine M. Pollock, Sandia National Laboratories

William Riddle, Software Design & Analysis

Martha Sloan, Michigan Technological University

Akinori Yonezawa, Tokyo Institute of Technology

Submissions

Send six double-spaced copies of articles and special-issue proposals to

Bruce D. Shriver, Vice President for Research, University of Southwestern

Louisiana, Drawer 42730, Lafayette, LA 70504-2730, phone (318) 231-5811.

For electronic submission, Shriver's addresses are b.shriver@compmail+ and shriver@usl.edu on Internet.

Staff

Editor and Publisher: True Seaborn

Assistant Publisher: Douglas L. Combs

Assistant to the Publisher: Pat Paulsen

Managing Editor: Marilyn Potes

Associate Editor: Nancy Hays

Staff Editors: Chuck Gormalde,

Steve H. Wilcox,

Bob Carlson

Ware Myers

Contributing Editor:

Art Director: Jay Simpson

Design/Production: Toni Van Buskirk, Janice Wright

Membership/Circulation: Christina Champion

Advertising Coordinators: Heidi Rex, Marian Tibayan

Magazine Advisory Committee

Jon T. Butler (chair), B. Chandrasekaran, Manuel D'Abreu,

James Farrell III, Joe Hootman, Sushil Jajodia, Ted Lewis,

True Seaborn, Bruce D. Shriver, John Staudhammer

Publications Board

Sallie Sheppard (chair), Dharma P. Agrawal, Victor Basili, P. Bruce Berra,

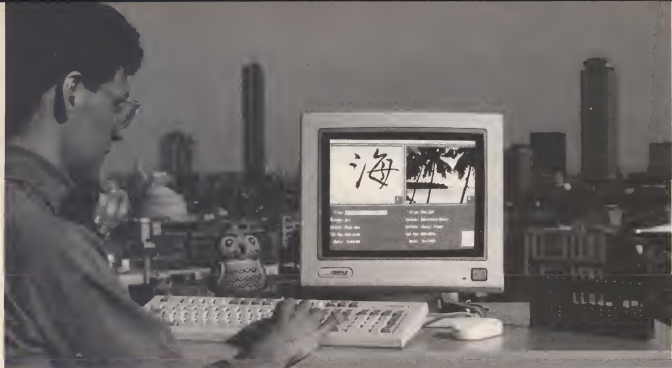
J. Richard Burke, Jon T. Butler, J.T. Cain, B. Chandrasekaran,

David Choy, James Cross, Manuel D'Abreu, James J. Farrell III,

Joe Hootman, Sushil Jajodia, Glen G. Langdon, Ted Lewis, Ming T. Liu,

C.V. Ramamoorthy, True Seaborn, Bruce D. Shriver, John Staudhammer,

Harold Stone, Steven L. Tanimoto, Joseph Urban, Ben Wah, Ron Williams



The joy of C-scape

The C-scape™ Interface Management System frees C programmers from the tedium of coding windows, menus, data validation, help, and text editing functions.

Moreover, C-scape is a joy to use. With C-scape's object-oriented design, you'll build more functional, more flexible, more portable, and more unique applications—and you'll have more fun doing it.

The industry standout. Many thousands of programmers have quit home-grown libraries and cumbersome, inflexible products for the pleasure of C-scape. The press agrees: "C-scape is by far the best . . . a joy to use," wrote IEEE Computer. PC Magazine chose C-scape to produce its Laboratory Benchmark Series 5.0 software because C-scape offers mouse support. Moreover, C-scape simultaneously combines text and graphics. And because C-scape makes it easy to create your own custom routines, major companies have selected C-scape as a standard for software development.

C-scape is built around an open architecture, so you can use it with data base management or other C libraries.

C-scape Features

Graphics. Combine high-resolution color graphics with text or menus.

Object-oriented. Add features and create reusable code modules.

Mouse. Use any standard mouse for fast screen control.

Portability. Write hardware independent code. Supports DOS, OS/2, UNIX, others. Autodetects Hercules, CGA, EGA, VGA.

Text editing. Create a full-featured text editor or pop-up note pad.

Field flexibility. Create masked, protected and marked fields with complete data validation. Use time, date, money, pop-up list, and many more functions, or create your own.

Windows. Choose from pop-up, tiled, bordered and exploding windows, with size and numbers limited only by RAM.

Menus. Choose from pop-up, pull-down, 123-style, or slug menus, or create your own.

Context-sensitive help. Link help messages to individual screens or fields. Cross reference messages to create hypertext-like help.

Screen design. Build any type of screen or form with the Look and Feel™ Screen Designer, then automatically convert it to C.

Screen flexibility. Call screens from files at run time or link them in.

And to port from MS-DOS or OS/2 to UNIX, just recompile.

Trial with a smile. C-scape is not only the most sophisticated, flexible and powerful interface system available, it's also the most friendly—and easiest to use. Try C-scape on a 30-day trial. It comes with a thorough manual, demo disk, sample programs with source code, an optional screen designer and code generator, access to a 24-hour bulletin board, and toll-free support. No royalties, runtime licenses, or runtime modules. After you register, you get complete library source code at no extra cost.

Call 800-233-3733 (617-491-7311 in Mass.) to try C-scape now. After the joy of C-scape, programming will never be the same.

MS-DOS, OS/2 \$399, library only; with Look & Feel \$499. UNIX, XENIX, Apollo, Sun, Stratus, others please call. Mastercard and Visa accepted.

OAKLAND

Oakland Group, Inc. 675 Massachusetts Ave., Cambridge, MA 02139 USA. FAX: 617-868-4440; Washington 206-746-8767; Benelux (02159)46814; Denmark (02)88 72 49; France (1)46 09 28 28; Germany/Austria/Switzerland (49)0127/5244; Norway (02)44 88 55; Sweden (013)124780; U.K. (0992)500019. C-scape and Look & Feel are trademarks of Oakland Group, Inc. MS-DOS and XENIX are trademarks of Microsoft Corp. OS/2 is a trademark of International Business Machines Corp. UNIX is a trademark of AT&T. HERCULES is a trademark of Hercules Computer Technology, Inc. Prices and terms subject to change. Photo by Jessica A. Boyatt. Kanji by Kaji Aso.

Reader Service Number 2



Guest Editors' Introduction

Recent Developments in Operating Systems

Joseph Boykin and Susan J. LoVerso

Encore Computer Corporation

*central processing
networks*

Prehistory, from an operating system point of view, was 1950. On the earliest digital computers, each user's job ran separately and had exclusive use of the computer for the duration of the program. The program communicated with I/O devices and managed other resources without assistance. When the program completed, an operator halted the computer and manually prepared the next job. The mid-1950s saw the dawn of batch processing and early operating systems which did little more than load programs and manage I/O devices. More general-purpose systems did not emerge until the mid-1960s.^{1,2} Many of these were multi-mode systems. They not only provided batch processing, but also included modes for time sharing and some real-time processing. These behemoths tried, within a single system, to provide every functional-ity to all users.⁴

Twenty-year history

Operating system refinement and consolidation came in the 1970s, when important concepts were developed, disappeared for a time, and then reappeared. For example, virtual memory, first shown on the Atlas system in 1959,³ did not become available in commercial systems until 1972, when IBM released it as part of its general product line.

For a time, many postulated that the operating system might disappear and that the user had no need to know of the central

services provided. The Apple Macintosh was cited as an example. It has no command line interpreter and no "system call" interface. The environment is simply a collection of library routines that provide access to the windowing system. But isn't this an operating system too?

Today, operating systems provide security, networking, distributed access, graphics, time-sharing, multiprocess, multithread, and multiprocessor capabilities but, unlike their ancestors, are not intended to be all things to all people within a single system. Like the computers on which they run, operating systems vary in size, complexity, and functionality to fulfill users' requirements. Some, like Digital Research's CP/M and Microsoft's MS-DOS, are little more than program loaders that also support a local disk drive. Others, such as DEC's VMS, Data General's AOS/VS, and AT&T's Unix, provide support for, and simultaneous access to, a large number of different peripheral devices. Sophisticated environments have been developed under these operating systems.

Dramatic changes in operating systems have resulted from enhancements to the hardware technology, experience with existing systems, development of new applications and needs, and, of course, imagination. Ten years ago, networking was an interesting concept with a few, unique interconnected systems; today, it's become a requirement. The combined developments of Ethernet and commodity microprocessors have provided a new

environment of high-performance, single-user graphical workstations connected over local area networks. LANs filled with personal workstations have created the demand for efficient use of computing and storage resources. Wide area networks, with the potential for worldwide connectivity, provide the possibility of global access to computing resources. Unfortunately, global agreement over the protocols used to access these resources seems unlikely, thereby placing new demands on operating systems to provide users with uniform and integrated data access.

Multiprocessor architectures have moved from experimental research labs into the mainstream of the computing world. The operating system, as the software foundation for these systems, is being asked to efficiently meet new requirements. Modern operating system implementations need to address such issues as load balancing across processors, gang scheduling, and algorithm locking.

Special-issue articles

Progress in the 20 short years of operating system history has been remarkable. The field has grown with the new, modern computer architectures. The operating system provides an ever-more-useful view of the system. This special issue of *Computer* looks at just a few of the recent developments. Its goal is to show where operating systems research has led us and where it might lead us in the future.

Early computer users placed the computer in the center of their world, with one or more large computers serving the needs of many. These systems were maintained by a dedicated staff. The advent of workstations shifted the focus to the user. The resulting problem was that users could no longer share data and now had the responsibility of administering their own systems — a task they were not trained to do. While no one wanted to eliminate workstations, it was recognized that a balance had to be struck. To accomplish this, we use few large systems as file and computation servers and leave the user interface on the workstation. The first article in this issue, "Scalable, Secure, and Highly Available Distributed File Access," by Mahadev Satyanarayanan, discusses the evolution of a distributed file system that provides highly available access to data through a combination of read/write replication and data caching.

Networking has become commonplace. What has not become common is the use of a single communications protocol. In some circumstances, this would be useful, but different protocols provide different functionality and capabilities. Most workstations provide only a single protocol family. Access to multiple protocol families, so that users can access different networks, will be an important requirement for the future. In the second article, "The x-Kernel: A Platform for Accessing Internet Resources," Larry Peterson, Norman Hutchinson, Sean O'Malley, and Herman Rao discuss an operating system designed to facilitate the implementation of multiple network protocols.

It has always amazed us that, after 20 years of wrestling with multiprocess operating systems, we are still trying to determine how to best schedule multiple processes. One reason is that the tasks we ask our systems to perform constantly change. Another is that computer architectures have also changed. Scheduling on a multiprocessor system should be different from scheduling on a uniprocessor — if not, we have not achieved optimal use of this new architecture. David L. Black's article, "Scheduling Support for Concurrency and Parallelism in the Mach Operating System," discusses scheduling techniques and policies for multiprocessor systems.

Advances in networking and workstations have not been independent activities. A natural outgrowth of these two developments has been an effort to make networked, multiple computer systems appear as a single entity. A new operating

system that implements this approach is Amoeba. Its design and implementation are discussed in "Amoeba: A Distributed Operating System for the 1990s," by Sape J. Mullender, Guido van Rossum, Andrew S. Tanenbaum, Robbert van Renesse, and Hans van Staveren.

These first four articles discuss the results of software development efforts. "Algorithms Implementing Distributed Shared Memory," by Michael Stumm and Songnian Zhou, mathematically analyzes the trade-offs of several distributed shared memory algorithms.

Multiprocessor systems have gained commercial success in the past few years. These systems, like all computers, continue to evolve. The final article, "Distributed Hierarchical Control for Parallel Processing" by Dror Feitelson and Larry Rudolph, discusses a new and unique approach to multiprocessing that allows dynamic repartitioning. ■

Acknowledgments

We would like to thank the many authors who submitted articles to this special issue. The quantity and quality of those submissions was more than we expected. We would also like to thank the many referees who donated countless hours to reading and commenting on papers. It is through their effort that issues such as this are possible.

We would also like to thank our colleagues and the management at Encore Computer. Editing this issue has taken a great deal of time — our management has been more patient with us than we had any right to expect. Our colleagues were called upon more than once to referee papers during the initial review period and to read and reread the final manuscripts.

Finally, we would like to thank Bruce Shriver, editor-in-chief of *Computer*, for his time, assistance, and suggestions, which enhanced the quality of this issue.

References

1. S. Rosen, "Electronic Computers: A Historical Survey," *Computing Surveys*, Vol. 1, No. 1, Mar. 1969, pp. 7-36.
2. R.F. Rosin, "Supervisory and Monitor Systems," *Computing Surveys*, Vol. 1, No. 1, Mar. 1969, pp. 37-54.
3. N. Weizer, "A History of Operating Systems," *Datamation*, Vol. 27, No. 1, Jan. 1981, pp. 119-126.
4. H.M. Deitel, *Introduction to Operating Systems*, Addison-Wesley, Reading, Mass., 1983.

5. J. Fotheringham, "Dynamic Storage Allocation in the Atlas Computer, Including an Automatic Use of a Backing Store," *Comm. ACM*, Vol. 4, No. 10, Oct. 1961, pp. 435-436.



Joseph Boykin is the manager of Mach Operating System Development with Encore Computer Corporation. The Mach OS group is responsible for the transition of Mach from a research prototype to a commercial product, as well as the use of Mach within research projects funded from sources both within and outside of Encore.

The Open Software Foundation will use Encore's implementation of Mach as the basis for the OSF/1 operating system. Encore's Mach OS group will provide consulting, design, and implementation support to OSF.

Boykin has held several leadership positions within the IEEE Computer Society. Currently, he is the society's treasurer and an Executive Committee member. He has chaired the Technical Committee on Operating Systems and the Workshop on Workstation Operating Systems.

Boykin holds both an MS in computer science and an MA in psychology. His graduate work was done at Ohio State and Pennsylvania State Universities.



Susan J. LoVerso has been a software engineer at Encore Computer Corporation since 1987 as a member of the Mach Operating System Development Group. Currently, she is working on parallelizing the 4.4BSD file system for the OSF/1 operating system.

LoVerso received her master's degree in computer science from the State University of New York at Buffalo. As a member of the Education Committee of the Women's Initiative for Technology Leadership in New England, she often speaks at local high schools about careers in the sciences and mathematics.

Readers can write to the guest editors at Encore Computer, 257 Cedar Hill St., Marlborough, MA 01752-3089. Their e-mail addresses are boykin@encore.com and sue@encore.com.

What's on your mind?

That's the question we've been asking our people since the beginning. And as a result of their valuable contributions, Xerox technology has revolutionized the way we work. To continue to develop technologies that will help business make impressive gains in productivity, we need the best talent the industry has to offer. If you have 5-7 years' experience developing embedded real-time software, you may qualify for a position with Team Xerox.

Systems Programmers

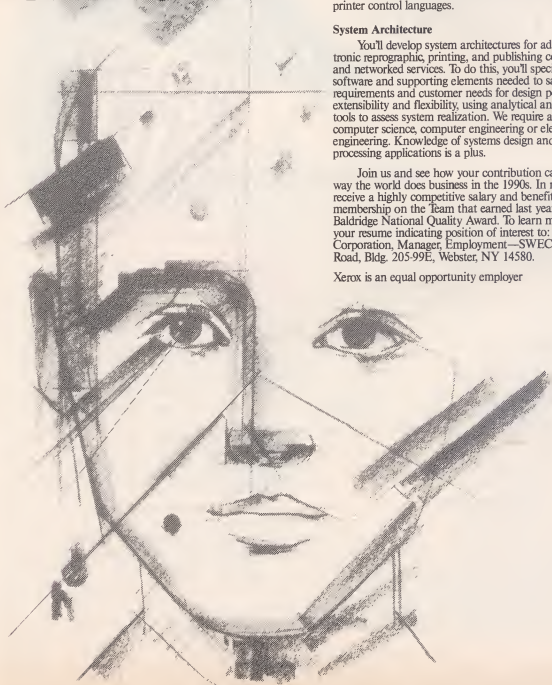
You'll design and develop software for advanced printing and electronic reprographic systems using our state-of-the-art highly typed, structured systems programming language (similar to ADA or Modula-2) in a state-of-the-art source language development/debug environment. You must have a BS/MS in electrical engineering, computer engineering or computer science. Familiarity with at least one of the following is a plus: structured design techniques, operating system internals, diagnostics, real-time performance analysis, software-hardware interfacing, distributed computing across LANs, page description languages and printer control languages.

System Architecture

You'll develop system architectures for advanced electronic reprographic, printing, and publishing components and networked services. To do this, you'll specify hardware, software and supporting elements needed to satisfy product requirements and customer needs for design performance, extensibility and flexibility, using analytical and simulation tools to assess system realization. We require an MS/PhD in computer science, computer engineering or electrical engineering. Knowledge of systems design and distributed processing applications is a plus.

Join us and see how your contribution can change the way the world does business in the 1990s. In return, you'll receive a highly competitive salary and benefits, and membership on the Team that earned last year's Malcolm Baldrige National Quality Award. To learn more, send your resume indicating position of interest to: Xerox Corporation, Manager, Employment—SWEC, 800 Phillips Road, Bldg. 205-99E, Webster, NY 14580.

Xerox is an equal opportunity employer



SUMMER 1990 USENIX TECHNICAL CONFERENCE & EXHIBITION

Anaheim, CA
June 11-15, 1990

TUTORIALS

The tutorial program provides a detailed examination of several UNIX topics. Two days of intensive, in-depth seminars will be presented by leading UNIX experts. Attendees needing to expand their knowledge will benefit from offerings in:

- Security
- User Interfaces
- Networking
- UNIX Internals
- Windowing Schemes
- Operating Systems
- Distributed Processing
- System Administration
- Writing Device Drivers

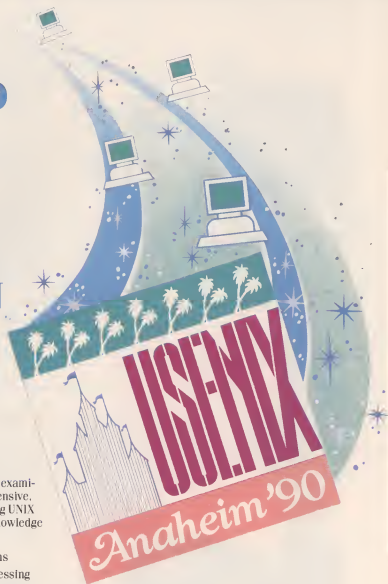
TECHNICAL EXHIBITION

The USENIX Exhibition allows hardware and software companies to display their latest, most advanced technical innovations to a highly focused end user community. For details on exhibiting, contact:

USENIX Exhibit Office,
5398 Manhattan Circle, Suite 200,
Boulder, CO 80303
Telephone (303) 499-2600,
FAX (303) 499-2608

TECHNICAL SESSIONS

The three day informative program will cover new and interesting work on a variety of technical issues including: Software Release Systems; User Interfaces, Windowing, Graphics; File Systems; Distributed Systems; UNIX Kernel Approaches; Compilers, Debuggers, Tools, Runtime Issues; Security and others.



ABOUT THE SPONSOR

The USENIX Association is a not-for-profit organization of those interested in UNIX and UNIX-like systems. It is dedicated to fostering and communicating the development of research and technological information and ideas pertaining to advanced computing systems.



*The Professional
and Technical
UNIX Association*

For complete conference details, call:
(714) 588-8649 or write:

USENIX Conference Office
22672 Lambert St., Suite 613
El Toro, CA 92630

UNIX is a registered Trademark of AT&T.

Scalable, Secure, and Highly Available Distributed File Access

Mahadev Satyanarayanan
Carnegie Mellon University

For the users of a distributed system to collaborate effectively, the ability to share data easily is vital. Over the last decade, distributed file systems based on the Unix model have been the subject of growing attention. They are now widely considered an effective means of sharing data in academic and research environments. This article presents a summary and historical perspective of work done by my colleagues, students, and I in designing and implementing such systems at Carnegie Mellon University.

This work began in 1983 in the context of Andrew, a joint project of CMU and IBM to develop a state-of-the-art computing facility for education and research at CMU. The project envisioned a dramatic increase in computing power made possible by the widespread deployment of powerful personal workstations. Our charter was to develop a mechanism that would enable the users of these workstations to collaborate and share data effectively. We decided to build a distributed file system for this purpose because it would provide the right balance between functionality and complexity for our usage environment.

It was clear from the outset that our distributed file system had to possess two critical attributes: It had to scale well, so that the system could grow to its antici-

Andrew and Coda are distributed Unix file systems that embody many of the recent advances in solving the problem of data sharing in large, physically dispersed workstation environments.

pated final size of over 5,000 workstations. It also had to be secure, so that users could be confident of the privacy of their data. Neither of these attributes is likely to be present in a design by accident, nor can it be added as an afterthought. Rather, each attribute must be treated as a fundamental constraint and given careful attention dur-

ing the design and implementation of a system.

Our design has evolved over time, resulting in three distinct versions of the Andrew file system, called AFS-1, AFS-2, and AFS-3. In this article "Andrew file system" or "Andrew" will be used as a collective term referring to all three versions.

As our user community became more dependent on Andrew, the availability of data in it became more important. Today, a single failure in Andrew can seriously inconvenience many users for significant periods. To address this problem, we began the design of an experimental file system called Coda in 1987. Intended for the same computing environment as Andrew, Coda retains Andrew's scalability and security characteristics while providing much higher availability.

The Andrew architecture

The Andrew computing paradigm is a synthesis of the best features of personal computing and timesharing. It combines the flexible and visually rich user interface available in personal computing with the ease of information exchange typical of

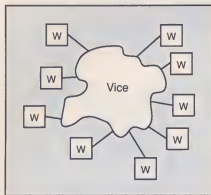


Figure 1. A high-level view of the Andrew architecture. The structure labeled "Vice" is a collection of trusted file servers and untrusted networks. The nodes labeled "W" are private or public workstations, or timesharing systems. Software in each such node makes the shared files in Vice appear as an integral part of that node's file system.

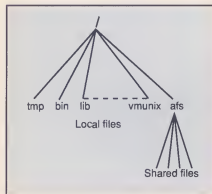


Figure 2. File system view at a workstation: how the shared files in Vice appear to a user. The subtree under the directory labeled "afs" is identical at all workstations. The other directories are local to each workstation. Symbolic links can be used to make local directories correspond to directories in Vice.

timesharing. A conceptual view of this model is shown in Figure 1.

The large, amoeba-like structure in the middle, called Vice, is the information-sharing backbone of the system. Although represented as a single entity, it actually consists of a collection of dedicated file servers and a complex local area network.

User computing cycles are provided by workstations running the Unix operating system.

Data sharing in Andrew is supported by a distributed file system that appears as a single large subtree of the local file system on each workstation. The only files outside the shared subtree are temporary files and files essential for workstation initialization. A process called Venus, running on each workstation, mediates shared file access. Venus finds files in Vice, caches them locally, and performs emulation of Unix file system semantics. Both Vice and Venus are invisible to workstation processes, which only see a Unix file system, one subtree of which is identical on all workstations. Processes on two different workstations can read and write files in this subtree just as if they were running on a single timesharing system. Figure 2 depicts the file system view seen by a workstation user.

Our experience with the Andrew architecture over the past six years has been positive. It is simple and easily understood by naive users, and it permits efficient implementation. It also offers a number of benefits that are particularly valuable on a large scale:

- *Data sharing is simplified.* A workstation with a small disk can potentially access any file in Andrew by name. Since the file system is location transparent, users do not have to remember the machines on which files are currently located or where files were created. System administrators can move files from one server to another without inconveniencing users, who are completely unaware of such a move.

- *User mobility is supported.* A user can walk to any workstation in the system and access any file in the shared name space. A user's workstation is personal only in the sense that he owns it.

- *System administration is easier.* Operations staff can focus on the relatively small number of servers, ignoring the more numerous and physically dispersed clients. Adding a new workstation involves merely connecting it to the network and assigning it an address.

- *Better security is possible.* The servers in Vice are physically secure and run trusted system software. No user programs are executed on servers. Encryption-based authentication and transmission are used to enforce the security of server-workstation communication. Although individuals may tamper with the hardware and software on their workstations, their malicious

actions cannot affect users at other workstations.

- *Client autonomy is improved.* Workstations can be turned off or physically relocated at any time without inconveniencing other users. Backup is needed only on the servers, since workstation disks are used merely as caches.

Scalability in Andrew

A scalable distributed system is one that can easily cope with the addition of users and sites, its growth involving minimal expense, performance degradation, and administrative complexity. We have achieved these goals in Andrew by reducing static bindings to a bare minimum and by maximizing the number of active clients that can be supported by a server. The following sections describe the evolution of our design strategies for scalability in Andrew.

AFS-1. AFS-1 was a prototype with the primary functions of validating the Andrew file system architecture and providing rapid feedback on key design decisions. Each server contained a local file system mirroring the structure of the shared file system. Vice file status information, such as access lists, was stored in shadow directories. If a file was not on a server, the search for its name would end in a stub directory that identified the server containing that file. Since server processes could not share memory, their only means of sharing data structures was via the local file system.

Clients cached pathname prefix information and used it to direct file requests to appropriate servers. The Vice-Venus interface named files by their full pathnames. There was no notion of a low-level name, such as the *inode* in Unix.

Venus used a pessimistic approach to maintaining cache coherence. All cached copies of files were considered suspect. Before using a cached file, Venus would contact Vice to verify that it had the latest version. Each open of a file thus resulted in at least one interaction with a server, even if the file was already in the cache and up to date.

For the most part, we were pleased with AFS-1. Almost every application was able to use Vice files without recompilation or relinking. There were minor areas of incompatibility with standard Unix semantics, but these were never serious enough to discourage users.

Design principles from Andrew and Coda

The design choices of Andrew and Coda were guided by a few simple principles. They were not specified a priori, but emerged in the course of our work. We share these principles and examples of their application in the hope that they will be useful to designers of other large-scale distributed systems. The principles should not be applied dogmatically but should be used to help crystallize thinking during the design process.

- **Workstations have the cycles to burn.** Whenever there is a choice between performing an operation on a workstation and performing it on a central resource, it is preferable to pick the workstation. This enhances the scalability of the design because it lessens the need to increase central resources as workstations are added.

The only functions performed by servers in Andrew and Coda are those critical to security, integrity, or location of data. Further, there is very little inter-server traffic. Pathname translation is done on clients rather than on servers in AFS-2, AFS-3, and Coda. The parallel update protocol in Coda depends on the client to directly update all AVSG members, rather than updating one of them and letting it relay the update.

- **Cache whenever possible.** Scalability, user mobility, and site autonomy motivate this principle. Caching reduces contention on centralized resources and transparently makes data available wherever it is being used.

AFS-1 cached files and location information. AFS-2 also cached directories, as do AFS-3 and Coda. Caching is the basis of disconnected operation in Coda.

- **Exploit file usage properties.** Knowledge of the nature of file accesses in real systems allows better design choices to be made. Files can often be grouped into a small number of easily identifiable classes that reflect their access and modification patterns. These class-specific properties provide an opportunity for independent optimization and, hence, improved performance.

Almost one-third of the file references in a typical Unix system are to temporary files. Since such files are seldom shared, Andrew and Coda make them part of the local name space. The executable files of system programs are often read but rarely written. AFS-2, AFS-3, and Coda therefore support read-only replication of these files to improve performance and availability. Coda's use of an optimistic replication strategy is based on the premise that sequential write sharing of user files is rare.

- **Minimize systemwide knowledge and change.** In a large distributed system, it is difficult to be aware at all times of the entire state of the system. It is also difficult to update distributed or replicated data structures consistently. The scalability of a design is enhanced if it rarely requires global information to be monitored or atomically updated.

Workstations in Andrew and Coda monitor only the status of servers from which they have cached data. They do not require any knowledge of the rest of the system. File location information on Andrew and Coda servers changes relatively rarely. Caching by Venus, rather than file location changes in Vice, is used to deal with movement of users.

Coda integrates server replication (a relatively heavyweight mechanism) with caching to improve availability without losing scalability. Knowledge of a caching site is confined to servers with call-backs for the caching site. Coda does

not depend on knowledge of systemwide topology, nor does it incorporate any algorithms requiring systemwide election or commitment.

Another instance of the application of this principle is the use of negative rights. Andrew provides rapid revocation by modifications of an access list at a single site rather than by systemwide change of a replicated protection database.

- **Trust the fewest possible entities.** A system whose security depends on the integrity of the fewest possible entities is more likely to remain secure as it grows.

Rather than trusting thousands of workstations, security in Andrew and Coda is predicated on the integrity of the much smaller number of Vice servers. The administrators of Vice need only ensure the physical security of these servers and the software they run. Responsibility for workstation integrity is delegated to the owner of each workstation. Andrew and Coda rely on end-to-end encryption rather than physical link security.

- **Batch if possible.** Grouping operations (and hence scalability) can improve throughput, although often at the cost of latency.

The transfer of files in large chunks in AFS-3 and in their entirety in AFS-1, AFS-2, and Coda is an instance of the application of this principle. More efficient network protocols can be used when data is transferred en masse rather than as individual pages. In Coda the second phase of the update protocol is deferred and batched. Latency is not increased in this case because control can be returned to application programs before the completion of the second phase.

AFS-1 was in use for about a year, from late 1984 to late 1985. At its peak usage, there were about 100 workstations and six servers. Performance was usually acceptable to about 20 active users per server. But sometimes a few intense users caused performance to degrade intolerably. The system turned out to be difficult to operate and maintain, especially because it provided

few tools to help system administrators. The embedding of file location information in stub directories made it hard to move user files between servers.

AFS-2. The design of AFS-2 was based on our experience with AFS-1 as well as on extensive performance analysis.¹ We retained the strategy of workstations caching

entire files from a collection of dedicated autonomous servers. But we made many changes in the realization of this architecture, especially in cache management, name resolution, communication, and server process structure.

A fundamental change in AFS-2 was the manner in which cache coherence was maintained. Instead of checking with a

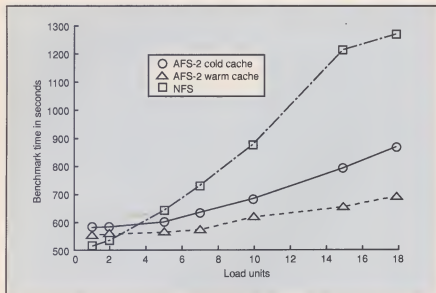


Figure 3. AFS-2 versus Sun NFS performance under load on identical client, server, and network hardware. A load unit consists of one client workstation running an instance of the Andrew benchmark. (Full details of the benchmark and experimental configuration can be found in Howard et al.,¹ from which this graph is adapted.) As the graph clearly indicates, the performance of AFS-2, even with a cold cache, degrades much more slowly than that of NFS.

server on each open, Venus now assumed that cache entries were valid unless otherwise notified. When a workstation cached a file or directory, the server promised to notify that workstation before allowing a modification by any other workstation. This promise, known as a *callback*, resulted in a considerable reduction in cache validation traffic.

Callback made it feasible for clients to cache directories and to translate pathnames locally. Without callbacks, the lookup of every component of a pathname would have generated a cache validation request. For reasons of integrity, directory modifications were made directly on servers, as in AFS-1. Each Vice file or directory in AFS-2 was identified by a unique fixed-length file identifier. Location information was contained in a slowly changing volume location database replicated on each server.

AFS-2 used a single process to service all clients of a server, thus reducing the context switching and paging overheads observed in AFS-1. A nonpreemptive lightweight process mechanism supported concurrency and provided a convenient programming abstraction on servers and clients. The RPC (remote procedure call)

mechanism in AFS-2, which was integrated with the lightweight process mechanism, supported a very large number of active clients and used an optimized bulk-transfer protocol for file transfer.

Besides the changes we made for performance, we also eliminated AFS-1's inflexible mapping of Vice files to server disk storage. This change was the basis of a number of mechanisms that improved system operability. Vice data in AFS-2 was organized in terms of a data-structuring primitive called a *volume*, a collection of files forming a partial subtree of the Vice name space. Volumes were glued together at mount points to form the complete name space. Venus transparently recognized and crossed mount points during name resolution.

Volumes were usually small enough to allow many volumes per server disk partition. Volumes formed the basis of disk quotas. Each system user was typically assigned a volume, and each volume was assigned a quota. Easily moved between servers by system administrators, a volume could be used (even for update) while it was being moved.

Read-only replication of volumes made it possible to provide increased availabil-

ity for frequently read but rarely updated files, such as system programs. The backup and restoration mechanism in AFS-2 also made use of volume primitives. To back up a volume, a read-only clone was first made. Then, an asynchronous mechanism transferred this frozen snapshot to a staging machine from which it was dumped to tape. To handle the common case of accidental deletion by users, the cloned backup volume of each user's files was made available as a read-only subtree of that user's home directory. Thus, users themselves could restore files within 24 hours by means of normal file operations.

AFS-2 was in use at CMU from late 1985 until mid-1989. Our experience confirmed that it was indeed an efficient and convenient system to use at large scale. Controlled experiments established that it performed better under load than other contemporary file systems.^{1,2} Figure 3 presents the results of one such experiment.

AFS-3. In 1988, work began on a new version of the Andrew file system called AFS-3. (For ease of exposition, all changes made after the AFS-2 release described in Howard et al.¹ are described here as pertaining to AFS-3. In reality, the transition from AFS-2 to AFS-3 was gradual.) The revision was initiated at CMU and has been continued since mid-1989 at Transarc Corporation, a commercial venture involving many of the original implementers of AFS-3. The revision was motivated by the need to provide decentralized system administration, by the desire to operate over wide area networks, and by the goal of using industry standards in the implementation.

AFS-3 supports multiple administrative cells, each with its own servers, workstations, system administrators, and users. Each cell is a completely autonomous Andrew environment, but a federation of cells can cooperate in presenting users with a uniform, seamless filename space. The ability to decompose a distributed system into cells is important at large scale because it allows administrative responsibility to be delegated along lines that parallel institutional boundaries. This makes for smooth and efficient system operation.

The RPC protocol used in AFS-3 provides good performance across local and wide area networks. In conjunction with the cell mechanism, this network capability has made possible shared access to a common, nationwide file system, distributed over nodes such as MIT, the University of Michigan, and Dartmouth, as well as CMU.

Venus has been moved into the Unix

Other contemporary distributed file systems

A testimonial to the importance of distributed file systems is the large number of efforts to build such systems in industry and academia. The following are some systems currently in use:

Sun NFS has been widely viewed as a de facto standard since its introduction in 1985. Portability and heterogeneity are the dominant considerations in its design. Although originally developed on Unix, it is now available for other operating systems such as MS-DOS.

Apollo Domain is a distributed workstation environment whose development began in the early 1980s. Since the system was originally intended for a close-knit team of col-

laboring individuals, scale was not a dominant design consideration. But large Apollo installations now exist.

IBM AIX-DS is a collection of distributed system services for the AIX operating system, a derivative of System V Unix. A distributed file system is the primary component of AIX-DS. Its goals include strict emulation of Unix semantics, ability to efficiently support databases, and ease of administering a wide range of installation configurations.

AT&T RFS is a distributed file system developed for System V Unix. Its most distinctive feature is precise emulation of local Unix semantics for remote files.

Sprite is an operating system for networked uniprocessor and multiprocessor workstations, designed at the University of California at Berkeley. The goals of the

Sprite file system include efficient use of large main memory caches, diskless operation, and strict Unix emulation.

Amoeba is a distributed operating system built by the Free University and CWI (Mathematics Center) in Amsterdam. The first version of the distributed file system used optimistic concurrency control. The current version provides simpler semantics and has high performance as its primary objective.

Echo is a distributed file system currently being implemented at the System Research Center of Digital Equipment Corporation. It uses a primary site replication scheme, with reelection in case the primary site fails.

Further reading

Surveys

Satyannarayanan, M., "A Survey of Distributed File Systems," in *Annual Review of Computer Science*, J.F. Traub et al., eds., Annual Reviews, Inc., Palo Alto, Calif., 1989.

Svobodova, L., "File Servers for Network-Based Distributed Systems," *ACM Computing Surveys*, Vol. 16, No. 4, Dec. 1984.

Individual systems

Amoeba
van Renesse, R., H. van Staveren, and A.S. Tanenbaum, "The Performance of the Amoeba Distributed Operating System,"

Software Practice and Experience, Vol. 19, No. 3, Mar. 1989.

Apollo Domain
Levine, P., "The Apollo Domain Distributed File System" in *Theory and Practice of Distributed Operating Systems*, Y. Paker, J.-T. Banatre, and M. Bozyigit, eds., NATO ASI Series, Springer-Verlag, 1987.

AT&T RFS
Rifkin, A.P., et al., "RFS Architectural Overview" *Proc. Summer Usenix Conf.*, Atlanta, 1986, pp. 248-259.

Echo
Hisgen, A., et al., "Availability and Consistency Trade-Offs in the Echo Distributed File System," *Proc. Second IEEE Workshop on*

Workstation Operating Systems, CS Press, Los Alamitos, Calif., Order No. 2003, Sept. 1989.

IBM AIX-DS
Sauer, C.H., et al., "RT PC Distributed Services Overview," *ACM Operating Systems Review*, Vol. 21, No. 3, July 1987, pp. 18-29.

Sprite
Ousterhout, J.K., et al., "The Sprite Network Operating System," *Computer*, Vol. 21, No. 2, Feb. 1988, pp. 23-36.

Sun NFS
Sandberg, R., et al., "Design and Implementation of the Sun Network File System," *Proc. Summer Usenix Conf.*, Portland, 1985, pp. 119-130.

kernel in order to use the *vnode* file intercept mechanism from Sun Microsystems, a de facto industry standard. The change also makes it possible for Venus to cache files in large chunks (currently 64 Kbytes) rather than in their entirety. This feature reduces file-open latency and allows a workstation to access files too large to fit on its local disk cache.

Security in Andrew

A consequence of large scale is that the casual attitude toward security typical of close-knit distributed environments is not

acceptable. Andrew provides mechanisms to enforce security, but we have taken care to ensure that these mechanisms do not inhibit legitimate use of the system. Of course, mechanisms alone cannot guarantee security; an installation also must follow proper administrative and operational procedures.

A fundamental question is who enforces security. Rather than trusting thousands of workstations, Andrew predicates security on the integrity of the much smaller number of Vice servers. No user software is ever run on servers. Workstations may be owned privately or located in public areas. Andrew assumes that the hardware and

software on workstations may be modified in arbitrary ways.

This section summarizes the main aspects of security in Andrew, pointing out the changes that occurred as the system evolved. These changes have been small compared to the changes for scalability. More details on security in Andrew can be found in an earlier work.³

Protection domain. The protection domain in Andrew is composed of *users* and *groups*. A user is an entity, usually a human, that can authenticate itself to Vice, be held responsible for its actions, and be charged for resource consumption. A

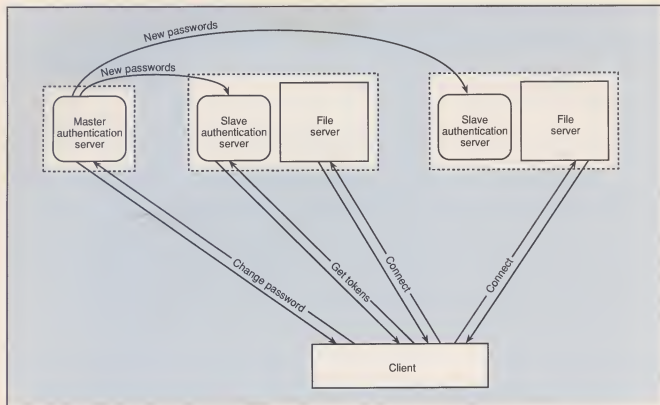


Figure 4. Major components and relationships involved in authentication in Andrew. Modifications such as password changes and additions of new users are made to the master authentication server, which distributes these changes to the slaves. When a user logs in, a client can obtain authentication tokens on the user's behalf from any slave authentication server. The client uses these tokens as needed to establish secure connections to file servers.

group is a set of other groups and users. Every group is associated with a unique user called its *owner*.

AFS-1 and AFS-2 supported *group inheritance*, with a user's privileges being the cumulative privileges of all the groups it belonged to, either directly or indirectly. Modifications of the protection domain were made off line by system administrators and typically were reflected in the system once a day. In AFS-3, modifications are made directly by users to a protection server that immediately reflects the changes in the system. To simplify the implementation of the protection server, the initial release of AFS-3 does not support group inheritance. This may change in the future because group inheritance conceptually simplifies management of the protection domain.

One group is distinguished by the name System:Administrators. Membership in this group endows special administrative privileges, including unrestricted access to any file in the system. The use of a System:Administrators group rather than a

pseudo-user (such as "root" in Unix systems) has the advantage that the actual identity of the user exercising special privileges is available for use in audit trails.

Authentication. The Andrew RPC mechanism provides support for secure, authenticated communication between mutually suspicious clients and servers, by using a variant of the Needham and Schroeder private key algorithm.⁴ When a user logs in on a workstation, his or her password is used to obtain tokens from an authentication server. These tokens are saved by Venus and used as needed to establish secure RPC connections to file servers on behalf of the user.

The level of indirection provided by tokens improves transparency and security. Venus can establish secure connections to file servers without users' having to supply a password each time a new server is contacted. Passwords do not have to be stored in the clear on workstations. Because tokens typically expire after 24 hours, the period during which lost tokens

can cause damage is limited.

As shown in Figure 4, there are multiple instances of the authentication server, each running on a trusted Vice machine. One of the authentication servers, the master, responds to updates by users and system administrators and asynchronously propagates the updates to other servers. The latter are slaves and only respond to queries. This design provides robustness by allowing users to log in as long as any slave or the master is accessible.

For reasons of standardization, the AFS-3 developers plan to adopt the Kerberos authentication system.⁵ Kerberos provides the functionality of the Andrew authentication mechanism and closely resembles it in design.

File system protection. Andrew uses an access list mechanism for file protection. The total rights specified for a user are the union of the rights specified for the user and for the groups he or she belongs to. Access lists are associated with directories rather than individual files. The reduction

in state obtained by this design decision provides conceptual simplicity that is valuable at large scale. An access list can specify *negative rights*. An entry in a negative rights list indicates denial of the specified rights, with denial overriding possession in case of conflict. Negative rights decouple the problems of rapid revocation and propagation of group membership information and are particularly valuable in a large distributed system.

Although Vice actually enforces protection on the basis of access lists, Venus superimposes an emulation of Unix protection semantics. The owner component of the Unix mode bits on a file indicate readability, writability, or executability. These bits, which indicate what can be done to the file rather than who can do it, are set and examined by Venus but ignored by Vice. The combination of access lists on directories and mode bits on files has proved to be an excellent compromise between protection at fine granularity, conceptual simplicity, and Unix compatibility.

Resource usage. A security violation in a distributed system can manifest itself as an unauthorized release or modification of information or as a denial of resources to legitimate users. Andrew's authentication and protection mechanisms guard against unauthorized release and modification of information. Although Andrew controls server disk usage through a per-volume quota mechanism, it does not control resources such as network bandwidth and server CPU cycles. In our experience, the absence of such controls has not proved to be a problem. What has been an occasional problem is the inconvenience to the owner of a workstation caused by the remote use of CPU cycles on that workstation. The paper on security in Andrew³ elaborates on this issue.

High availability in Coda

The Coda file system, a descendant of AFS-2, is substantially more resilient to server and network failures. The ideal that Coda strives for is constant data availability, allowing a user to continue working regardless of failures elsewhere in the system. Coda provides users with the benefits of a shared data repository but allows them to rely entirely on local resources when that repository is partially or totally inaccessible.

When network partitions occur, Coda allows data to be updated in each partition but detects and confines conflicting updates as soon as possible after their occurrence. It also provides mechanisms to help users recover from such conflicts.

A related goal of Coda is to gracefully integrate the use of portable computers. At present, users manually copy relevant files from Vice, use the machine while isolated from the network, and manually copy updated files back to Vice upon reconnection. These users are effectively performing manual caching of files with write-back on reconnection. If one views the disconnection from Vice as a deliberately induced failure, it is clear that a mechanism for supporting portable machines in isolation is also a mechanism for fault tolerance.

By providing the ability to move seamlessly between zones of normal and disconnected operation, Coda may simplify the use of cordless network technologies such as cellular telephone, packet radio, or infrared communication in distributed file systems. Although such technologies provide client mobility, they often have intrinsic limitations such as short-range, inability to operate inside steel-framed buildings, or line-of-sight constraints. These shortcomings are reduced in significance if clients are capable of temporary autonomous operation.

The design of Coda was presented in detail in a recent paper.⁶ A large subset of the design has been implemented, and work is in progress to complete the implementation. One can sit down at a Coda workstation today and execute Unix applications without recompilation or relinking. Execution continues transparently when contact with a server is lost due to a crash or network failure. In the absence of failures, using a Coda workstation feels no

different from using an AFS-2 workstation.

Design overview. The Coda design retains key features of AFS-2 that contribute to scalability and security:

- Clients cache entire files on their local disks. From the perspective of Coda, whole-file transfer also offers a degree of intrinsic resiliency. Once a file is cached and open at a client, it is immune to server and network failures. Caching on local disks is also consistent with our goal of supporting portable machines.

- Cache coherence is maintained by the use of callbacks.

- Clients dynamically find files on servers and cache location information.

- Token-based authentication and end-to-end encryption are used as the basis of security.

Coda provides failure resiliency through two distinct mechanisms. It uses *server replication*, or the storing of copies of files at multiple servers, to provide a highly available shared storage repository. When no server can be contacted, the client resorts to *disconnected operation*, a mode of execution in which the client relies solely on cached data. Neither mechanism is adequate alone. While server replication increases the availability of all shared data, it does not help if all servers fail or if all are inaccessible due to a network failure adjacent to a client. On the other hand, permanent disconnected operation is infeasible. The disk storage capacity of a client is a small fraction of the total shared data. Permanent disconnected operation is also inconsistent with the Andrew model of treating each client's disk merely as a cache. Key advantages of the Andrew architecture, namely mobility and a user's ability to treat any workstation as his or her own, are lost.

From a user's perspective, transitions between these complementary mechanisms are seamless. A client relies on server replication as long as it remains in contact with at least one server. It treats disconnected operation as a measure of last resort and reverts to normal operation at the earliest opportunity. A portable client that is isolated from the network is effectively operating in disconnected mode.

When network partitions occur, Coda allows data to be updated in each partition but detects and confines conflicting updates as soon as possible after their occurrence. It also provides mechanisms to help

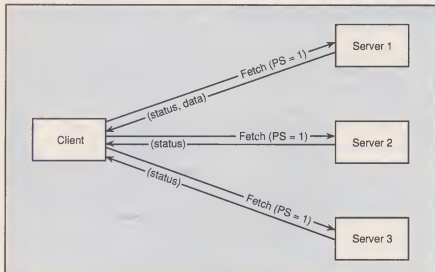


Figure 5. Servicing a cache miss in Coda: the events that follow from a cache miss at the client. Both data and status are fetched from Server 1, which is the preferred server (PS). Only status is fetched from Server 2 and Server 3. The calls to all three servers occur in parallel.

users recover from such conflicts. This strategy is *optimistic*, in contrast to a *pessimistic* strategy that would preserve strict consistency by disallowing updates in all but one partition. We chose an optimistic strategy for two reasons: First, we saw no clean way to support disconnected opera-

tion with a pessimistic strategy. Second, it is widely believed that sequential write sharing between users is relatively infrequent in Unix environments, so conflicting updates are likely to be rare.

Coda provides a scalable and highly available approximation of Unix seman-

tics. We arrived at this semantics on the basis of our positive experience with AFS-2. In the absence of failures, Coda and AFS-2 semantics are identical. On open, the latest copy of a file in the system is cached from Vice. Read and write operations are made to the cached copy. On close, the modified file is propagated to Vice. Future opens anywhere in the system will see the new copy of the file. In the presence of failures, Coda and AFS-2 semantics differ. An open or close in AFS-2 would fail if the server responsible for the file was inaccessible. In Coda, an open fails only on a cache miss during disconnected operation or if a conflict is detected. A close fails only if a conflict is detected.

Server replication. The unit of replication in Coda is a volume. A *replicated volume* consists of several physical volumes, or replicas, that are managed as one logical volume by the system. Individual replicas are not normally visible to users. The set of servers with replicas of a volume constitutes its *volume storage group* (VSG). The degree of replication and the identity of the replication sites are specified when a volume is created. Although these parameters can be changed later, we do not anticipate such changes to be frequent. For every volume from which it has cached data, Venus keeps track of the subset of the VSG that is currently accessible. This subset is called the *accessible VSG* (AVSG). Different clients may have different AVSGs for the same volume at a given instant. Venus performs periodic probes to detect shrinking or enlargement of the AVSGs from which it has cached data. These probes are relatively infrequent, occurring once every 10 minutes in our current implementation.

Coda integrates server replication with caching, using a variant of the read-one, write-all strategy. This variant can be characterized as read-one-data, read-all-status, write-all. In the common case of a cache hit on valid data, Venus avoids contacting the servers altogether. When servicing a cache miss, Venus obtains data from one member of its AVSG, known as the *preferred server*. The PS can be chosen at random or on the basis of performance criteria such as physical proximity, server load, or server CPU power. Although data is transferred only from one server, Venus contacts the other servers to collect their versions and other status information. Venus uses this information to check whether the accessible replicas are equivalent. If the replicas are in conflict, the

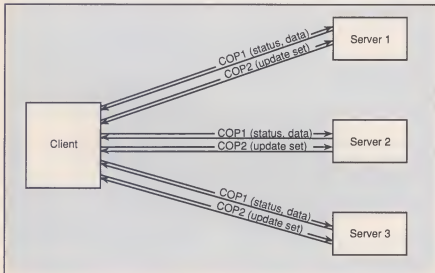


Figure 6. A store operation in Coda: the two phases of the Coda update protocol. In the first phase, COP1, the three servers are sent new status and data in parallel. In the later asynchronous phase, COP2, the update set is sent to these servers. COP2 also occurs in parallel and can be piggybacked on the next COP1 to these servers.

system call that triggered the cache miss is aborted. If the replicas are not in conflict but some replicas are stale, the AVSG is notified asynchronously that a refresh is necessary. In the special case where the data on the PS is stale, a new PS is selected and the fetch is repeated. The message exchange in the normal case, where there is no conflict and the PS has the latest copy of the data, is illustrated in Figure 5. A call-back is established with the PS as a side-effect of successfully fetching the data.

When a file is closed after modification, it is transferred to all members of the AVSG. This approach is simple to implement and maximizes the probability that every replication site has current data at all times. Server CPU load is minimized because the burden of data propagation is on the client rather than the servers. This in turn improves scalability, since the server CPU is the bottleneck in many distributed file systems. Operations that update directories, such as creating a new directory or removing a file, are also written through to all AVSG members.

Because its replication scheme is optimistic, Coda checks for existing conflicts on each server operation. The update protocol also guarantees eventual detection of new conflicts caused by the update. This protocol consists of two phases, COP1 and COP2, where COP stands for Coda *optimistic protocol*. The first phase performs the semantic part of the operation, such as transferring file contents, making a directory entry, or changing an access list. Each server verifies that its copy does not conflict with the client's copy before performing the update. The second phase distributes to the servers a data structure called the *update set*, which summarizes the client's knowledge of who performed the COP1 operation. The update set maintains the version information used in conflict detection. Figure 6 illustrates the message exchange in a store operation (which corresponds to a file close).

Two protocol optimizations improve performance: First, latency is reduced by Venus's returning control to the user after completion of COP1 and performing COP2 asynchronously. Second, network and server CPU loads can be reduced by Venus's piggybacking the asynchronous COP2 messages on subsequent COP1 calls to the same VSG.

At present, a server performs no explicit remote actions upon recovery from a crash. Rather, it depends on clients to notify it of stale or conflicting data. Although this

lazy strategy does not violate Coda's consistency guarantees, it does increase the chances of a future conflict. A better approach, which we plan to adopt in the future, is for a recovering server to contact other servers to bring itself up to date.

Each server operation in Coda typically involves multiple servers. If the operation were carried out sequentially, latency would increase significantly. Venus therefore communicates with replication sites in parallel, using a parallel RPC mechanism. This mechanism has been extended to use hardware multicast support, if available, to reduce the latency and network load caused by shipping large files to multiple sites. Shipping a large file to three servers in our current implementation typically takes about 10 percent longer than shipping it to one server.

Operation latency is usually a major concern with replication schemes, but server replication in Coda has worked well. Controlled experiments on identical client and server hardware show that under light loads Coda's performance is within five percent of the performance of Andrew's current release. Thus, the cost of replication is primarily the storage cost for additional replicas at the servers. The current implementation of Coda does not perform quite as well under heavy load. Our measurements indicate specific areas for improvement, and we are confident that these changes will result in an implementation with significantly better performance under load.

Disconnected operation. Disconnected operation begins at a Coda workstation when no member of a VSG is accessible. Clients view it as a temporary state and revert to normal operation at the earliest opportunity. A client may be operating in disconnected mode with respect to some volumes but not others. Disconnected operation is transparent to a user unless a cache miss occurs. A cache miss normally aborts the system call that triggered the reference, but it is possible to arrange for such system calls to block. Return to normal operation is also transparent, unless a conflict is detected.

To reduce the chances of a cache miss during disconnected operation, Coda allows a user to specify a prioritized list of files and directories that Venus should strive to retain in the cache. Objects of the highest priority level are "sticky" and must be retained at all times. As long as the local disk is large enough to accommodate all sticky files and directories, the user can

always access them. Since it is often difficult to know exactly what file references are generated by a certain set of high-level user actions, Coda provides the ability for a user to bracket a sequence of high-level actions and for Venus to note the file references generated during these actions. The implementer of an application can also provide a list of files that should be made sticky for the application to work when disconnected.

Disconnected operation with respect to a particular volume ends when Venus reestablishes connection with any member of the volume's VSG. Reconnection results from a successful probe—either one of Venus's periodic probes or one manually induced by a user-level command. The transition from disconnected operation invokes a process of reintegration. For each cached file or directory that has been created, deleted, or modified on the client during disconnected operation, Venus executes a sequence of update operations to make AVSG replicas identical to the cached copy. Reintegration proceeds top-down, from the root to the leaves of modified subtrees.

Update operations during reintegration may fail for one of two reasons. First, there may be no authentication tokens that Venus can use to communicate securely with AVSG members. Users whose tokens expire during disconnected operation may forestall reintegration until they have reacquired valid tokens to minimize this possibility. Second, conflicts may be detected. Given our model in which servers rather than clients are dependable storage repositories, we felt that the proper approach to handling both of these situations was to find a temporary home on servers for the data in question and to rely on a user to resolve the problem later.

The temporary repository is realized as a *covolume* for every replica of every volume in Coda. Covolumes are similar in spirit to lost-found directories in Unix. They have a flat name space derived from the original low-level identifiers of the objects they contain. Covolumes are not directly visible to users but are accessed indirectly through a repair tool as described in the next section. *Migrate* is the operation that transfers a file or directory from a workstation to a covolume. Having a covolume per replica allows us to perform migration immediately upon reintegration failure rather than waiting for connection to a particular site. The storage overhead of this approach is usually small since a covolume is almost always empty.

Mechanisms for building distributed file systems

Although there is considerable diversity in the manner in which distributed file systems are put together, all the current ones are built from a surprisingly small number of basic mechanisms. This sidebar presents the most important of these mechanisms and examples of how different systems have used them.

- **Mount points.** The mount mechanism in Unix enables the gluing together of filename spaces to provide applications with a single, seamless, hierarchically structured name space. In a distributed Unix file system, the mount mechanism provides a natural hook on which to hang a remote subtree.

There are two different ways to use the mechanism. The simpler approach is used by systems such as Sun NFS, in which each client individually mounts subtrees from servers. Although this approach is easier to implement, it has the disadvantage that the shared name space may not be identical at all clients. Further, movement of files from one server to another requires each client to unmount and remount the affected subtree. In practice, systems that use this approach have usually had to provide auxiliary mechanisms (such as the Yellow Pages and Automounter in Sun NFS) to automate and centralize mounts.

The alternative approach is to embed mount information in the data stored in the file servers. Andrew and Coda, for example, use mount points embedded in volumes. Sprite uses remote links for a similar purpose. This approach makes it relatively simple to ensure that all clients see the same shared filename space at all times.

- **Client caching.** The caching of data at clients is undoubtedly the architectural feature that contributes most to performance in a distributed file system. Every distributed file system in serious use today uses some form of caching. Even AT&T's RFS, which initially avoided caching in the interests of strict Unix emulation, now uses it. In most systems, clients maintain the cache in their main memory. Andrew and Coda, in contrast, cache on the local disk, with a further level of caching in main memory.

A key issue in caching is the size of the cached units of data. Most distributed file systems cache individual pages of files. Coda, Amoeba, AFS-1, and

AFS-2 cache entire files. AFS-3 and most other file systems cache portions of a file.

Cache validation can be done in two ways. One approach is for the client to contact the server for validation. The alternative approach, used in AFS-2, AFS-3, Coda, AIX-DS, and Echo, is to have the server notify clients when cached data is about to be rendered stale. Although more complex to implement, the latter approach can produce substantial reductions in client-server traffic.

Existing systems use a wide spectrum of approaches in propagating modifications from client to server. AIX-DS usually propagates changes to the server only when the file is explicitly flushed. Andrew and Coda propagate changes when a file is closed after writing. Sprite delays propagation until dirty cache pages have to be reclaimed or for a maximum of 30 seconds. Deferred propagation improves performance since data is often overwritten, but it increases the possibility of server data being stale due to a client crash.

- **Hints.** In the context of distributed systems, a hint is a piece of information that can substantially improve performance if correct but has no semantically negative consequence if erroneous. For maximum performance benefit, a hint should nearly always be correct. By caching hints, one can obtain substantial performance benefits without incurring the cost of maintaining cache consistency. Only information that is self-validating upon use is amenable to this strategy. One cannot, for instance, treat file data as a hint because the use of a cached copy of the data will not reveal whether it is current or stale.

Hints are most often used for file location information in distributed file systems. Sprite, for instance, caches mappings of pathname prefixes to servers. Similarly, Andrew and Coda cache individual entries from the volume location database. Apollo Domain uses a more elaborate location scheme incorporating a hint manager.

- **Bulk data transfer.** Network communication overhead caused by protocol processing typically accounts for a major portion of the latency in a distributed file system. Transferring data in bulk reduces this overhead by amortizing fixed protocol overheads over many consecutive pages of a file. For effectiveness, bulk transfer protocols depend on spatial lo-

cality of reference within files.

The degree to which bulk transfer is exploited varies from system to system. Amoeba, Andrew, and Coda are critically dependent on it. Sun NFS and Sprite exploit bulk transfer by using very large packet sizes, typically 8 kilobytes. Bulk transfer protocols will increase in importance as distributed file systems spread across networks of wider geographic area and thus have greater inherent latency.

- **Encryption.** Encryption is an indispensable building block for enforcing security in a distributed system. It is used for remote authentication and for preventing unauthorized release and modification of data transmissions. The national standard DES (data encryption standard) is the most commonly used form of private-key encryption. The seminal work of Needham and Schroeder on the use of encryption for authentication is the basis of all current security mechanisms in distributed file systems.

Authentication can be performed with private or public keys. In the private-key schemes used by Kerberos and Andrew, a physically secure authentication server maintains a list of user passwords in the clear. In contrast, the public-key scheme used by Sun NFS maintains a publicly readable database of authentication keys encrypted with user passwords. The latter approach has the attractive characteristic that physical security of the authentication server is unnecessary. Its major drawback is that public-key encryption is computationally more expensive.

- **Replication.** Replication of data at multiple servers is the primary mechanism for providing high availability. The more recent file systems such as Coda and Echo provide read-write replication of data. Amoeba supports read-write replication at the directory level because files are immutable in that system. Although read-write replication is well understood theoretically, little experience of its use exists as yet.

More experience has been gathered with read-only data replication, which is supported by systems such as Sun NFS and Andrew. Though suitable only for files that change relatively rarely, it is valuable because many critical files (such as system binaries) possess this property.



NEED TO KEEP UP TO DATE WITH COMPUTER SCIENCE?

Computer Science is a rapidly-developing field of research. You need high-quality scientific information about the latest developments and research results in order to keep up to date.

North-Holland offers you an extensive high-level publication programme with journals and books covering a wide range of subject areas, such as: Information Systems, Computer Communications, Distributed Systems & Telecommunications, Theoretical Computer Science & Database Theory, Artificial Intelligence, Expert Systems, Knowledge Engineering, Computer Architecture, Parallel and Supercomputing, Computer Performance, Simulation and Modelling, Industrial Automation, Computer Graphics, Pattern Recognition, Computers in Education, Computers & Security.

North-Holland's Computer Science Catalogue 1990 contains full details of our latest books and our range of journals. Write for your personal copy of the catalogue by completing the coupon!



NORTH-HOLLAND PUBLICATIONS

North-Holland is an imprint of Elsevier Science Publishers BV,
one of the world's largest scientific publishers.

COUPON

COMP/M

Send this coupon - or a photocopy - to:

North-Holland
(An Imprint of Elsevier Science Publishers BV)
Attn.: Petra van der Meer
P.O. Box 103
1000 AC Amsterdam
The Netherlands

Customers in the USA/Canada:

Elsevier Science Publishing Co. Inc.
P.O. Box 882
Madison Square Station
New York, NY 10159, USA

☐ Yes, please send me my personal copy of the
Computer Science Catalogue 1990

Name _____
Address _____
Country _____

We are currently in the midst of implementing disconnected operation. Although we are confident of our ability to support short-term disconnected operation (for a few minutes or hours), it remains to be seen whether long-term disconnected operation (for days or weeks) is feasible. Our concerns center on the overall size of the working set and on the predictive power of our caching strategies. Our own experience, and that of others, suggests that a cache of several tens of megabytes should be adequate for a typical disconnection of less than a day. Less obvious is whether any anticipatory caching strategy can, with a reasonable cache size, provide the near-perfect hit rates required for long-term disconnected operation.

Conflict resolution. When a conflict is detected, Coda first attempts to resolve it automatically. Since Unix files are typed byte streams, there is no information to automate their resolution. A directory, on the other hand, is an object whose semantics is completely known and whose resolution can often be automated. For example, partitioned creation of uniquely named files in the same directory can be handled automatically by selectively replaying the missing creates. If automated resolution is not possible, Coda marks all accessible replicas of the object inconsistent and moves them to their respective covolumes. This ensures damage containment because normal operations on these replicas will subsequently fail.

Coda provides a repair tool to assist users in manually resolving conflicts. It uses a special interface to Venus so that requests from the tool are distinguishable from normal file system requests. This enables the tool to overwrite inconsistent files and to perform directory operations on inconsistent directories. The tool has evolved along with the rest of our system. Three generations of the tool are described here: the tool for the currently implemented system, the one we are working on at present, and a successor that will incorporate the current wish list.

In the first-generation tool, inconsistent files and directories are marked in conflict but are not moved to covolumes. Disconnected operation is not supported because there is nowhere to migrate objects to. When the tool is invoked on a given object, it mounts the accessible replicas of the object's volume in a scratch area of the name space. The user can then use normal Unix applications to inspect the replicas. The replicas are mounted in read-only

The general problem of sharing information effectively in large distributed systems is far from being solved. The next decade poses many challenges and promises to be a fertile and exciting period for researchers in this area.

mode so that the user cannot inadvertently alter anything. When the user has decided on a fix (such as selecting the version in one of the replicas to be the new permanent one), the tool performs the fix and cleans up the workspace.

The second-generation tool supports disconnected operation because it knows about covolumes. Inconsistent objects are immediately moved to the associated covolume when the inconsistency is detected. When the tool is invoked, it constructs a temporary workspace and mounts, read-only, the covolumes as well as the replicas of the object. As before, the user can navigate through the replicas. However, names of inconsistent objects now correspond to objects in the associated covolumes. The tool applies the fix and cleans up the workspace as the first-generation tool does.

The primary refinement provided by the third-generation tool will be a considerably simplified user interface. Venus, in conjunction with the tool, will present the illusion of an in-place "explosion" of inconsistent objects into their distinct versions. Invocation of the tool will put Venus in a mode whereby inconsistent objects can be viewed within the existing name space. In this mode, Venus will map an inconsistent file or directory into a read-only directory with the same name as the original. This directory will be populated with entries translated by Venus to the versions in the various covolumes. The tool will handle the fix phase of the repair in the same way as the second-generation tool.

Throughout the evolution of the Andrew and Coda file systems, the underlying model of computation has remained unchanged. A small collection of trusted servers jointly provides a shared data repository for a much larger number of untrusted workstations. The system design facilitates incremental growth by the addition of users and workstations. The security of the system is not contingent upon the integrity of the workstations or of the network.

The problems of scalability, security, and availability will continue to be important as distributed file systems grow in size. In addition, three other problems will be of fundamental importance to the broader goal of effective data sharing in large distributed systems. These are the problems of heterogeneity, access to diverse types of data, and rapid search.

As a distributed system grows, it tends to become more heterogeneous. Coping with heterogeneity is inherently difficult because of the presence of multiple computational environments, each with its own notions of file naming and functionality. Since few general principles are applicable, the idiosyncrasies of each new system have to be accommodated by ad hoc mechanisms. Unfortunately, heterogeneity cannot be ignored since it is likely to be a chronic problem.

Alternative models of data are likely to become more important in the future. Relational databases are already in widespread use in certain application domains. Speech, music, images, and video are examples of other forms of data that the repositories of the future will have to store and retrieve. We presently have little knowledge of how to share such diverse types of data in large-scale distributed systems.

Finding data in a large distributed file system is already difficult. As distributed storage repositories grow larger and store more diverse types of data, the problem of searching for relevant information will become acute. This is another area where we have barely scratched the surface.

In conclusion, we have made much progress in the design and implementation of distributed file systems over the last decade. Andrew and Coda embody many of the key advances made during this period. But the general problem of sharing information effectively in large distributed systems is far from being solved. The next decade poses many challenges and promises to be a fertile and exciting period for researchers in this area. ■

Acknowledgments

The Andrew file system was built by the File System Group of the Information Technology Center at Carnegie Mellon University. The membership of this group over time has included Ted Anderson, Sailesh Chutani, John Howard, Michael Kazar, Sherri Meneses Nichols, David Nichols, Mahadev Satyanarayanan, Robert Sidebotham, Michael West, and Edward Zayas. Contributions to the early design of Andrew were also made by David Gifford and Alfred Spector.

Coda is being built in the School of Computer Science at Carnegie Mellon University. Contributors to Coda include James Kistler, Puneet Kumar, Maria Okasaki, Mahadev Satyanarayanan, Ellen Siegel, Walter Smith, and David Steere.

James Kistler assisted in writing this article.

This research was supported by the National Science Foundation (contract No. CCR-8657907), Defense Advanced Research Projects Agency (order No. 4976, contract No. F33615-87-C-1499), IBM Corporation (faculty development award, graduate fellowship, and the Andrew project), and Digital Equipment Corporation (equipment grant). The views and conclusions in this article are those of the author and do not represent the official policies of the funding agencies or of Carnegie Mellon University.

References

1. J.H. Howard et al., "Scale and Performance in a Distributed File System," *ACM Trans. Computer Systems*, Vol. 6, No. 1, Feb. 1988, pp. 51-81.
2. M.N. Nelson, B.B. Welch, and J.K. Ousterhout, "Caching in the Sprite Network File System," *ACM Trans. Computer Systems*, Vol. 6, No. 1, Feb. 1988, pp. 134-154.
3. M. Satyanarayanan, "Integrating Security in a Large Distributed System," *ACM Trans. Computer Systems*, Vol. 7, No. 3, Aug. 1989, pp. 247-280.
4. R.M. Needham and M.D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Comm. ACM*, Vol. 21, No. 12, Dec. 1978, pp. 993-998.
5. J.G. Steiner, C. Neumann, and J.I. Schiller, "Kerberos: An Authentication Service for Open Network Systems," *Proc. Usenix Conf.*, Dallas, Texas, Feb. 1988, pp. 191-202.
6. M. Satyanarayanan et al., "Coda: A Highly Available File System for a Distributed Workstation Environment," *IEEE Trans. Computers*, Vol. 39, No. 4, Apr. 1990, pp. 447-459.



Mahadev Satyanarayanan is an associate professor of computer science at Carnegie Mellon University. His research addresses the general problem of sharing access to information in large-scale distributed systems. He was one of the principal architects and implementers of the Andrew file system and currently leads the Coda project. His work on Scylla explored access to relational databases in a distributed workstation environment. His previous research included the design of the CMU-CFS file system, measurement and analysis of file usage data, and the modeling of storage systems.

Satyanarayanan received the PhD in computer science from Carnegie Mellon in 1983, after receiving a bachelor's degree in electrical engineering and a master's degree in computer science from the Indian Institute of Technology, Madras. He is a member of the IEEE, the IEEE Computer Society, ACM, and Sigma Xi, and has been a consultant to industry and government. He was named a Presidential Young Investigator by the National Science Foundation in 1987.

Readers can write to Satyanarayanan at the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213-3890.

At GTE's Computer and Intelligent Systems Laboratory, we are creating opportunities to apply new ideas to research and development projects for improved information and telecommunication systems. By expanding our scope and responsibility, we can better support GTE's telecommunications businesses. And our activities create challenges for individuals at the MS/PhD level in Computer Science. Join us as we continue to make history in telecommunications technology.

Our present areas of interest include:

Distributed Operating Systems

We are currently growing a group that is conducting research in distributed operating systems and distributed transaction systems. These systems will unify a network of cooperating autonomous, heterogeneous processors and replicated databases. Issues concern not only the tradeoffs between network and distributed operating systems, but the interoperability of software components implemented on a mix of platforms and languages; included are such topics as transport, access, application, distributed control and control migration. Research is conducted using synthesis and prototyping with emphasis on architectural and applicability issues. We are looking for individuals at all levels of experience, however, we require a PhD in Computer Science along with a familiarity with the various current approaches to the design of distributed systems. Significant experience with at least one such system is highly desirable.

Intelligent Database Systems

Our Distributed Object Management (DOM) project is conducting research into interconnectivity and intelligent interoperability among heterogeneous computer systems. We require PhD-level researchers at all levels with a minimum of 2 years' experience in databases, operating systems, distributed systems or artificial intelligence.

Research Scientist Information Retrieval

We are extending the state-of-the-art in Information Retrieval systems, primarily through the development of a multi-media system that includes full-text retrieval, graphics, pictures and other structurable information; algorithms and their implementation for document modeling; and the user-interface to the retrieval system itself. Research prototypes will be developed in C and X-windows on multiple operating system platforms. We require a minimum of a MS in Computer Science, and 3 years' programming with C under various operating systems such as VMS, UNIX, MS-DOS, and Macintosh.

GTE Laboratories offers attractive facilities located in a quiet, wooded setting just outside of Boston, as well as a highly competitive salary and benefits package. We invite you to send a resume to Vanessa Stern, GTE Laboratories, Inc., Box IEEECS90, 40 Sylvan Road, Waltham, MA 02254. An equal opportunity employer, M/F/H/V.



Laboratories

THE POWER IS ON

19 Reasons Why We're Proud to Work for Amdahl



The results of the 1989 Datapro Survey of Mainframe Computer Users are a good indication of the pride we take in our work at Amdahl. Once again, our people swept the field, earning the highest ratings in 19 of 25 performance categories.

- We ranked first in "overall customer satisfaction" for the second year running.
- We ranked first in mainframe reliability, maintenance, service and technical support.
- We were the only company to notch a perfect score—100%—when mainframe users were asked if their system met all of their expectations.
- We were also the only company to score a perfect 100% when users were asked if they'd recommend their system to a fellow user.

To earn such accolades, year after year, we put the greatest emphasis on recruiting and developing a staff of the highest caliber. Our projects demand new ideas, techniques and solutions. So, as large as we are, with more than \$2 billion in annual sales worldwide, we structure our teams for originality, visibility and the encouragement of personal and professional growth.

We also recognize outstanding performance, rewarding our people for meeting our high level of expectation. Our competitive salaries and comprehensive benefits are first class. And, as the results of the Datapro survey show, our customers recognize our efforts in at least 19 different ways.

PROCESSOR PRODUCTS. This division designs and develops Amdahl's large-scale, high-performance mainframe systems and product software. These systems implement industry-leading packaging and design innovations and are compatible with the industry-standard System/370-architecture. This division hires individuals with all levels of experience and a BSEE/ MSEE/PhDEE or BSCS/MSCS/PhDCS, or equivalent, in the following areas:

- Product Software & Diagnostics
 - MVS, VM Operating Systems Internals
 - 370 Assembling
 - Diagnostics
 - UNIX* Development
 - C and REXX Programming
 - Macrocode Development
- Design Automation
 - Simulation
 - Timing Analysis
 - Test Generation
 - Design Verification
 - System Administration
- System Architecture
 - Computer Architecture
 - Interface Development
- Computer Development
 - Circuit Design
 - Packaging Technology
 - Logic Design
- S/W Engineering
 - Application Development
 - MVS Programming
 - Relational Database Development
 - 370 Assembler & C

STORAGE PRODUCTS. Storage Products develops and markets innovative, high-performance peripherals for on-line data storage. The 6100 storage controller adheres to industry standards, while providing greater capacity and flexibility than the competition. The 6380 family of disk storage devices offer more—more performance, more capacity, more features—in a durable, compact cabinet.

- Software Development
 - MVS/XA Internals
 - 370 Assembly Language
 - I/O Diagnostic Programming
- Firmware Development
 - Microcode Development
 - Controller Development
 - Assembly Language Development
 - Firmware Diagnostics

TECHNICAL MARKETING Our UNIX Systems Group documents benchmark requirements to our Amdahl Performance Evaluation Center (AMPEC) and monitors all completion time frames. We also evaluate competitive products and provide configuration results for publication. We require internal/external knowledge of UNIX (UTS**), with an emphasis on configuration, performance and networking implementation, to perform in the following areas:

- Systems Engineering
- UNIX Internals

For Storage Products and Technical Marketing positions, contact Dottie DeSelle at 800-538-8460, ext. 75981, or send your resume to her at Mail Stop 300.

The satisfaction of exceptional challenge awaits you. Send your resume to Amdahl Corporation, Employment Department 4-3, P.O. Box 3470, Mail Stop 300, Sunnyvale, CA 94088-3470. Principals only, please.

Amdahl Corporation is proud to be an equal opportunity employer through affirmative action.

- *UNIX is a registered trademark of AT&T.
- **UTS is a trademark of Amdahl Corporation.

Contact Ranell Durgan at 800-538-8460, ext. 66216 or send your resume to her at Mail Stop 300.

amdahl



The x-kernel: A Platform for Accessing Internet Resources

Larry Peterson, Norman Hutchinson, Sean O'Malley, and Herman Rao

University of Arizona

The x-kernel is an experimental operating system for personal workstations that allows uniform access to resources throughout a nationwide internet: an interconnection of networks similar to the TCP/IP Internet. This network is also called the National Research and Education Network (NREN).

This focus on national rather than local resources is central to the x-kernel's design and sets it apart from other experimental operating systems. A current trend in operating system design is to make a collection of workstations appear to function as a single time-sharing system. In such a system, workstations on the same local area network are tightly coupled: they run the same kernel and the same special-purpose protocol suite. Such designs include the V system and its versatile message-transaction protocol,¹ the Sprite network operating system and its remote procedure call (RPC) protocol,² and the Amoeba system and its RPC protocol.³

However, workstation users connected to national networks have access to significantly more resources than are available on local networks. The NREN, for example, connects researchers throughout the country to databases, file systems, directory services, supercomputers, and other specialized hardware. The availability of resources on such national networks will grow as network bandwidth and connectivity increase.



**The x-kernel gives
workstation users
uniform access to
resources across local or
wide area networks. The
operating system is
general, efficient, and
easy to program.**

The key to making such diverse resources available is to implement the necessary communication protocols on the user's workstation. For example, a workstation must implement the NFS protocol to access a file in a Sun network file system⁴ and the AFS protocol to access a file in an Andrew file system.⁵ Similarly, a workstation must implement the Sun RPC protocol to invoke a service provided by the Sun operating system⁶ and the Sprite RPC protocol to invoke a service provided by the Sprite operating system. In general, the more protocols on the user's workstation, the more resources the user can access. Of course, user-level software must

also be available to take advantage of these resources.

While a single file-access protocol, RPC protocol, and directory protocol would simplify matters, it is not likely that a single protocol suite can provide access to a national network's resources in the same way that a distributed operating system can mandate a single protocol suite for a local area network. One reason for the diversity of protocols is that loosely coupled national networks like the NREN consist of autonomous organizations, each free to define its own protocols for accessing its own resources. Also, performance constraints often dictate different protocols for accessing remote resources than those for accessing local resources. For example, most file access protocols on local area networks (with low latency) transfer data one page at a time, while most file access protocols on wide area networks (with high latency) transfer complete files.

Thus, our driving philosophy views the workstation as a portal through which users access both local and remote network resources. The x-kernel realizes this philosophy by providing an infrastructure that makes it easy to implement network protocols.⁷ This infrastructure is general enough to support a wide variety of protocols, yet efficient enough that no protocol suffers a serious performance penalty. In contrast, many existing operating systems are either built around a single protocol or do not

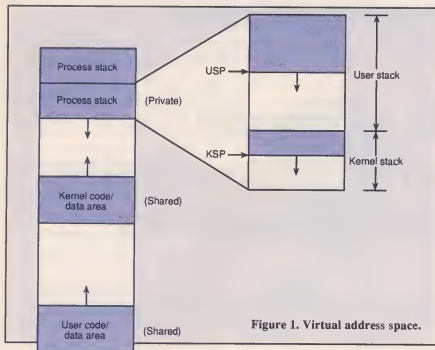


Figure 1. Virtual address space.

provide enough explicit infrastructure and support to encourage protocol implementations. In fact, our experience is that most operating systems actually discourage new protocols by making it difficult to implement them.

The x-kernel supports a library of protocols, and it accesses different resources with different protocol combinations. However, if we had stopped here, users would face the impossible task of selecting the appropriate protocol to access a given resource. Users also need a convenient interface and useful application programs that take advantage of the protocols in the

kernel. Toward this end, we have built two user-level systems on top of the x-kernel that give users an integrated and uniform interface to resources. These two systems — a file system and a command interpreter — hide differences among the underlying protocols.

The x-kernel

The x-kernel currently runs on Sun-3 workstations, and we plan to port it to other hardware platforms. It includes components that manage processes, memory, and

communication. The process and memory managers resemble those in other operating systems: multiple address spaces are supported, multiple light-weight processes can execute in each address space, and processes within an address space synchronize using kernel-supported semaphores.

The communication manager is the kernel's most novel aspect. It provides an object-oriented infrastructure for composing protocols and a collection of powerful tools for implementing tasks common to all protocols. We designed the communication manager to balance generality and efficiency. Generality means that the x-kernel can implement a wide variety of network protocols; it does not mandate a particular protocol suite. Rather than being limited to a single RPC protocol, the x-kernel can have any number of RPC protocols; we have implemented three. Efficiency means none of the protocols in the x-kernel suffer severe performance penalties. In fact, protocols run as fast in the x-kernel as they do in protocol-specific operating systems.

Process and memory management. Because network protocols are central to the x-kernel's design, we designed both the process and memory managers around the needs of the communication manager. One of the most performance-critical aspects of any workstation operating system is its ability to pass control and data efficiently between the kernel and user programs. This is because data messages sent by user programs must be passed to the kernel for transmission over the network, and packets that arrive on the network must eventu-

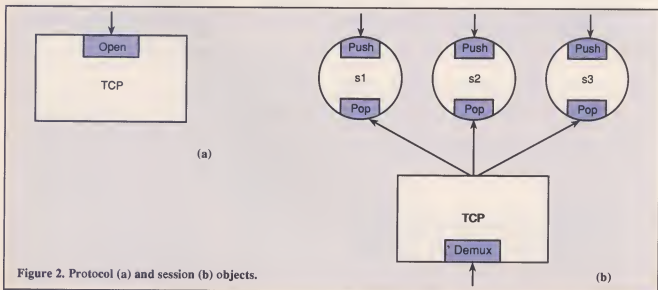


Figure 2. Protocol (a) and session (b) objects.

ally be passed to a user program.

Consider the structure of a virtual address space, as shown in Figure 1. The processes share a user area and a kernel area, but each process has its own private stack. The user and kernel areas each consist of code, static data, and dynamic data. Each private stack contains a user stack and a kernel stack. Note that the kernel area is mapped to each virtual address space; that is, a single copy of the kernel is shared among all the address spaces. On a Sun-3 with a 28-bit virtual address space, the kernel reserves 16 megabytes of virtual address space for its code and data area and the stack area, leaving 240 megabytes of address space for the user program. The kernel code and data typically occupy less than 500 kilobytes of physical memory.

A process can execute in either user or kernel mode. A process that executes in user mode — called a user process — uses the code and data in the user area and the user portion of its private stack. Likewise, a process that executes in kernel mode — a kernel process — uses the code and data in the kernel area and the kernel portion of its stack. Note that a kernel process has access to both the kernel and user areas, while a user process only has access to the data in the user area; the kernel code and data are protected by the memory-management hardware.

The x-kernel is symmetric in the sense that a user process can change to a kernel process (this corresponds to a typical system call) and a kernel process can invoke a user-level procedure (we call this an up-call). When a user process invokes a kernel system call, a hardware trap makes the process execute kernel code and use the kernel-stack pointer. User data remains accessible because the new kernel process executes in the same virtual address space.

When a kernel process invokes a user-level procedure, it executes a preamble routine that sets up an initial activation record in the user stack, pushes the arguments to the procedure onto the user stack, and starts using the user-stack pointer. The process can then only access data in the user area. Rather than copy large arguments (such as data messages) into user space, the kernel simply puts them in memory pages and maps the pages to the user's data area. Because there is a danger of the user procedure not returning (as in an infinite loop), the kernel limits the number of outstanding calls to each user address space.

Thus, control and data can pass in either direction without expensive data copying.

Uniform protocol interface

`session = open(high_level_protocol, low_level_protocol, participants)`

A high-level protocol calls a low-level protocol to actively open a session with the given participant set.

`openenable(high_level_protocol, low_level_protocol, participants)`

A high-level protocol calls a low-level protocol to passively open (enable the opening of) a session that contains the given participant set.

`session = opendone(low_level_protocol, high_level_protocol, participants)`

A low-level protocol calls back the high-level protocol to complete the passive open and reports the participant set for the established session.

`demux(session, msg)`

A session of some low-level protocol passes a message up to a high-level protocol.

`controlprotl(protocol, opcode, buffer, length)`

Perform a control operation on some protocol; used to set and get parameters.

`push(session, msg)`

Pass a message down to a session.

`pop(session, msg)`

Pass a message up to a session.

`close(session)`

Close a session.

`controlsessn(session, opcode, buffer, length)`

Perform a control operation on some session; used to set and get parameters.

Crossing the user/kernel boundary on a Sun-3/75 takes 20 microseconds from user to kernel and 254 microseconds from kernel to user. The latter takes an order of magnitude longer because the kernel must implement the boundary-crossing mechanism in software. There is no hardware analog of the system trap for going from kernel mode to user mode.

Communication management. The communication manager takes advantage of the process and memory managers to support any protocol suite required by application programs. At its heart lies an object-oriented infrastructure for implementing and composing protocols, based on two abstract communication objects: protocols and sessions. Both objects export a well-defined set of operations implemented by each network protocol the kernel will support. Having all protocols implement the same set of operations is useful because it provides a common interface to dissimilar protocols, simplifying the task of composing protocols.

For example, Figure 2a depicts a TCP protocol object that supports a uniform

operation called Open. When a high-level entity wants to send a message using TCP, it invokes TCP's Open operation to create a TCP session. Figure 2b shows the TCP protocol object and three associated session objects. Each TCP session provides the protocol interpreter and data structures to maintain the local state of a TCP connection. Messages pass through a TCP session in two directions. Outgoing messages are sent down by invoking the session's Push operation, and incoming messages are sent up by invoking its Pop operation.

The TCP protocol object also supports a demultiplexer operation called Demux. When a low-level protocol (such as IP) finishes processing a message and is ready to hand it off to TCP, it invokes TCP's Demux operation, which selects the TCP session responsible for processing the message. The Demux operation then invokes that session's Pop operation. In other words, the TCP protocol object switches each incoming message to one of the TCP session objects. This switching decision is based on information in the message, such as a TCP port number. The sidebar titled "Uniform protocol interface" gives the

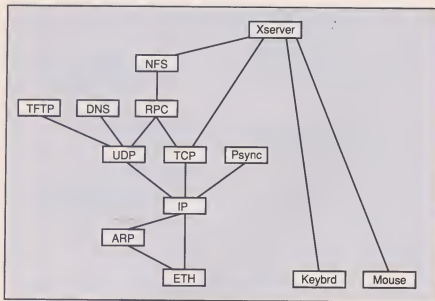


Figure 3. Protocol suite.

complete set of the x-kernel's uniform protocol operations.

Thus, each network protocol is encapsulated in a single protocol object and a collection of session objects. To compose a collection of protocols, each protocol invokes the uniform operations exported by the other protocols on which it depends. Figure 3 shows a collection of protocol objects that could be supported by a particular configuration of the x-kernel. For simplicity, only the protocol objects are shown.

Note that the TCP protocol in this example is directly used by two other kernel protocols: RPC and Xserver. TCP is also directly available to user-level programs. To keep the protocol interface clean and eliminate special cases, all protocols assume that another protocol sits above them in the protocol dependency graph. No protocol is considered a "top most" protocol directly invoked by a user program. Therefore, to invoke an operation on a kernel-level protocol, a user-level program must invoke a Createprotocol operation to disguise itself as a protocol object. The Createprotocol arguments are pointers to various user-level procedures that implement operations, such as Demux, that a low-level protocol might expect a high-level protocol to support. If TCP has a message to pass up to the user, it invokes the user's Demux procedure. The Createprotocol operation returns a handle for the newly created "user" protocol. The user program uses this handle to identify itself to kernel-level protocols. For ex-

ample, when a user program invokes TCP's Open operation to create a TCP session, it passes this handle as the first argument.

Given a configuration of protocols such as the one in Figure 3, the communication manager uses the process and memory managers in the following way. When a message arrives at a network device, a kernel process shepherds it up through a sequence of protocol and session objects. If the message eventually reaches the user/kernel boundary—if a kernel-level protocol wants to invoke a user's Demux operation—the shepherd process crosses the boundary and continues executing in user mode. When a user process generates a message, the process crosses the user/kernel boundary and shepherds the message down through a sequence of protocol and session objects in the kernel.

For efficiency, the x-kernel does not create a new shepherd process for each message received. Instead, the kernel keeps a pool of warm-started shepherd processes that it assigns to incoming messages. The overhead to dispatch one of these processes is 135 microseconds. This overhead is small enough, relative to the rate at which packets are delivered by the Ethernet, that the x-kernel drops less than one in a million packets when configured with the Ethernet controller in promiscuous mode (that is, when accepting packets addressed to any host on the network).

Thus, rather than associate a process with each protocol, where an interprocess communication mechanism passes net-

work packets from one protocol to another, the x-kernel associates a message with each network packet. Responsibility for processing the packet passes from one protocol to another by having the first protocol invoke a procedure (operation) exported by the second. The x-kernel's process-per-message design makes it trivial to avoid a process switch between every protocol level. In fact, when a message does not encounter contention for resources, a message can be sent or received with no process switches. When resource contention does occur, the shepherd process blocks on a semaphore.

In addition to the uniform protocol operations, the x-kernel has a library of support routines that provides efficient solutions to programming tasks common to all network protocols. Specifically, the support library includes a message manager, a map manager, and an event manager. Each protocol's implementation of the uniform operations—such as Open, Demux, Push, and Pop—uses these three managers.

The message manager provides a framework for manipulating the data, headers, and trailers that make up network packets. The message manager defines a single abstract data type: the message. By casting packets in this abstract message type and applying a well-defined set of operations to the abstract data type, protocols can add headers and trailers to messages, strip headers and trailers from messages, fragment messages into multiple packets, reassemble multiple packets into a single message, insert discrete packets into a continuous stream, and break a continuous stream into discrete packets. More importantly, the message manager can support all of these operations without unnecessary data copying. The message manager is a critical aspect of the x-kernel's performance. It is described in more detail in the sidebar titled "The message abstraction."

The map manager is a hash table optimized to efficiently compare variable-length external identifiers. Protocols use the map manager to switch packets from lower level protocols to the appropriate session object. This switching function maps an external identifier from the packet header (such as addresses or port numbers) to an internal identifier for the session that handles the message. Such mappings are defined when network connections are established (that is, in the Open operation) and used each time a packet is received from a lower level protocol (that is, in Demux). The map manager maintains a set of bindings of one identifier to another,

and it supports operations for adding new bindings, removing bindings from the set, and mapping one identifier to another relative to a set of bindings.

The event manager provides an "alarm clock" for protocols. Network protocols are often described, if not implemented, as finite automata driven by external events such as arriving messages and expiring timers. The event manager lets a protocol specify a timer event as a procedure to be called at some future time. By registering a procedure with the event manager, proto-

cols can time out and act on messages that have not been acknowledged.

Finally, to configure a kernel that contains a particular protocol suite — such as the one in Figure 3 — a user merely specifies the protocol graph using a simple graph description language. A composer utility reads the description and links the appropriate object modules.

Performance. The *x*-kernel supports efficient implementations of a wide variety of protocols. Table 1 reports the la-

Table 1. Latency of various protocols.

Protocol	<i>x</i> -kernel (msec)	Other Systems (msec)
UDP	2.0	5.4 (Unix)
		7.5 (Sprite)
TCP	3.3	6.1 (Unix)
		11.0 (Mach)
Sun RPC	4.0	9.2 (Unix)
Sprite RPC	1.7	2.6 (Sprite)

The message abstraction

The *x*-kernel message manager implements a simple abstraction: Messages are merely arbitrarily sized strings of bytes. The difficulty in implementing this abstraction lies in ensuring that an arbitrary sequence of operations performed on a message during its lifetime — that is, as the message passes through multiple protocol layers — can execute as efficiently as possible (without unnecessary data copying).

As an outgoing message makes its way through the kernel, each of several protocols' Push operations can add a header to the message (prepend a string) or fragment the message into multiple shorter packets (divide a string into substrings). Similarly, while processing an incoming message, each of several protocols' Pop operations will strip the header from the message (delete a string's prefix) or reassemble message fragments (concatenate two strings). In addition, any of these protocols might save references to portions of the message for future use (retransmission).

To support these kinds of manipulations, the *x*-kernel's message manager provides the following operations:

`top = msg_push(msg, len)`

Add *len* bytes to the beginning of *msg* and return a pointer to the beginning of the message. Used to add a header.

`msg_pop(msg, len)`

Discard *len* bytes from the beginning of *msg*. Used to strip a header.

`top = msg_top(msg, len)`

Return a pointer to the beginning of *msg*, ensuring that at least the first *len* bytes are contiguous. Used to inspect the header of a message.

`msg3 = msg_join(msg1, msg2)`

Concatenate *msg₁* and *msg₂*. Used to recombine two message fragments.

`msg2 = msg_break(msg1, len)`

Split *msg₁* into two pieces, the first of which contains *len* bytes. Used to fragment a message.

`msg2 = msg_save(msg1)`

Save a (logical) copy of *msg₁* for later use.

`msg_free(msg)`

Free a previously saved copy of a message.

`len = msg_len(msg)`

Return the length of *msg*.

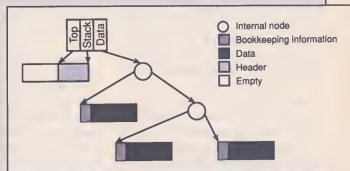
There are additional operations to create messages in a variety of ways, to linearize messages before delivering them

to the user, and so on, but none of them are critical to protocol performance in the *x*-kernel.

To efficiently implement all these operations, the message manager represents each message as a directed acyclic graph of data buffers, augmented with a stack data structure as depicted in the accompanying figure. The data buffers that make up the leaves of the graph are allocated from the kernel's data heap. The bytes stored in a message can be retrieved by concatenating the data in the stack with the data found in a preorder traversal of the graph structure.

The stack at the beginning of a message is optimized to support the `msg_push` and `msg_pop` operations because copying a small header is much more efficient than allocating storage for graph nodes (which would be required if the graph were used for headers). Pushing header information onto a message's stack involves adjusting the stack pointer and copying the data; popping header information off a message's stack involves only adjusting the stack pointer. Because the stack is of a fixed size, it is only used to hold headers and very small messages; larger messages are represented using the graph structure.

The graph structure is optimized to efficiently implement the `msg_join`, `msg_break`, `msg_save`, and `msg_free` operations. The graph nodes are reference counted, so a logical copy of a message is made by increasing a reference count. A message is freed by decrementing a reference count. Joining two messages involves allocating a new graph node that points to the root nodes of the two original messages. Likewise, breaking a message requires allocating an internal node that points to a substring of the original message. Bytes are not copied in any of these operations, which is critical to performance, since any message that requires fragmentation is fundamentally long to begin with.



tency of several protocols in the x-kernel; that is, the time to exchange a pair of messages between two processes. For comparison, we also give the performance of each protocol in one or more other operating systems. In the table, Unix refers to SunOS 4.0 (a variant of 4.3 BSD Unix). All times are user-to-user except for Sprite RPC, which is kernel-to-kernel.

We measured the x-kernel and Unix results on a pair of Sun-3/75s connected by a lightly loaded 10-megabit/second Ethernet. The numbers were derived through two levels of aggregation. Each experiment exchanged a 1-byte message 10,000 times between a pair of user processes and recorded the elapsed time every 1,000 round-trips. Table 1 shows the average of these 10 elapsed times. Although we do not report the standard deviations of the various experiments, they were small. The Sprite and Mach latency times were reported by the systems' implementors; both measure the time to exchange a small message between two user processes. The Sprite number is for the same hardware configuration. The Mach number is for Sun-3/60s rather than 3/75s. (A Sun-3/60 is 33 percent faster than a 3/75.)

The performance differences between the x-kernel and other operating systems are instructive. First, Unix' protocol performance is hampered by overhead imposed by the socket abstraction. Measurements of the individual protocols in isolation show the two systems' performance to be nearly the same: for UDP, the x-kernel and Unix times are 0.11 millisecond and 0.25 millisecond, respectively; for TCP, 1.41 milliseconds and 1.30 milliseconds, respectively. Moreover, these comparisons are a fair measure of the operating system overhead rather than of the protocol implementations. In the case of UDP, the protocol is simple enough to leave little room for variation between the implementations. In the case of TCP, the x-kernel implementation was directly derived from the Unix implementation; it does not take advantage of any special techniques. The results suggest that explicitly structured protocols in the x-kernel perform competitively with ad hoc implementations in Unix.

Second, Sprite is an example of a system designed around its own custom RPC protocol. Other protocols such as UDP are implemented outside the kernel and so cannot take advantage of kernel facilities such as the buffer manager. Mach is another example of a system that, at least in

Table 2. RPC throughput.

Method	Throughput (Kbytes/s)
Kernel-to-Kernel	860
User-to-User (Remapping)	748
User-to-User (Copying)	575

principle, moves network code outside the kernel. However, Mach's TCP time in Table 1 comes from version 2.5, which still implements the network code within the kernel. We believe TCP will perform less efficiently when exported out of the kernel, as will be the case in version 3.0.

Third, for Sprite RPC we did not control for differences in implementation techniques. However, the x-kernel, with its general architecture, can still perform competitively with a kernel built around a specific protocol.

Because the x-kernel remaps pages containing large messages to the destination user-address space, it can also achieve high user-to-user throughput. Consider Table 2. The first row reports the throughput between two kernels when an RPC protocol was used to exchange 8-kilobyte messages (the page size on a Sun-3). The second and third rows give the effective user-to-user throughput for the same RPC protocol using page remap and byte copying, respectively, to get the messages across the user/kernel boundary. Remapping pages imposes a 13 percent penalty on the kernel-to-kernel throughput, while byte copying imposes a 34 percent penalty. Note that the remapping strategy is more complicated than simply manipulating page tables; the kernel must also catch the data part of the incoming packets that make up the 8-kilobyte message in successive offsets of some page.

Related systems. The x-kernel is by no means the only operating system kernel that explicitly supports communication protocols. For example, Berkeley Unix supports the socket abstraction,⁸ and System V Unix supports the stream abstraction.⁹ The key strength of the x-kernel, however, is that it is the only system we know of built around the communication system rather than the other way around.

As a consequence, the x-kernel's abstractions and support library are more general, uniform, and efficient than those in other systems.

Berkeley Unix actually has three different protocol interfaces: driver/protocol, protocol/socket, and socket/application. Only the top-most protocols are encapsulated in the socket abstraction. In contrast, the same x-kernel abstractions are used by all protocols, including application-level protocols, network protocols, and device drivers. Also, the overhead imposed on each protocol by the x-kernel's uniform protocol interface is negligible, whereas the overhead of sockets is the dominant factor in Berkeley Unix' protocol performance.

For System V Unix, the stream abstraction was originally designed for character (terminal) I/O. The system was extended by adding multiplexer objects to accommodate the complexity of network protocols. In contrast, the x-kernel abstractions were designed explicitly for network protocols. Although the System V and x-kernel communication infrastructures are certainly similar, the differences in origin influenced several details of each system's design. First, the x-kernel supports distinct paths for data and control, while System V folds data and control into a single path. Second, the x-kernel is general enough to support both synchronous and asynchronous interaction between protocol layers, while System V supports only asynchronous interaction. This means the programmer must "step outside" the stream abstraction to implement a synchronous protocol like RPC. Finally, the x-kernel supports a process-per-message paradigm, while System V supports a process-per-protocol paradigm.

Also note that not all operating systems provide explicit support for implementing network protocols. At one extreme, systems like the V-kernel mandate a particular protocol or protocol suite. Because such operating systems support only a fixed set of known protocols, the protocols can be embedded in the kernel without being encapsulated in a general protocol abstraction. At the other extreme, systems such as Mach¹⁰ move responsibility for implementing protocols out of the kernel. Such systems view each protocol as an application implemented on top of the kernel; that is, they provide no protocol-specific infrastructure. (As noted earlier, Mach's current release has not yet moved the network code out of the kernel.)

Logical file system

The x-kernel lets workstation users access any resources on whatever local area and wide area networks the workstation is connected to. In particular, the x-kernel supports access to multiple file systems. It provides a uniform interface by implementing a logical file system. The system is logical in the sense that it depends entirely on one or more existing physical file systems; it does not implement any storage of its own. In other words, the x-kernel provides only a directory service that maps file names to a location where the file can be found. Such a file system is appropriate for a workstation that depends on one or more file servers. A workstation with a local disk also requires a local file system, which is treated like any other physical file system and is not considered part of the logical file system.

The file system has two unique features. First, it is defined on a per-user basis: Each user constructs a private file system out of various existing file systems. Because each x-kernel file system is associated with a user instead of a machine, each user sees the same file system regardless of the workstation used. Figure 4 illustrates a file system associated with a particular user, where the dotted lines denote the partitioning of the logical hierarchy into a set of physical file systems. Each physical file system is said to be "mounted" in the user's private file system.

A user's private file system usually behaves just like a Unix file system. The x-kernel supports the same set of file system calls as does Unix: Open, Read, Write, Lseek, and so on. On the other hand, because the user is responsible for creating his or her private file system, the x-kernel provides two new operations to mount other file systems in the user's private hierarchy and to read what systems are mounted in that hierarchy. These operations, called Lmkdir for "logical make directory" and Lreadir for "logical read directory," are described below. Thus, the network is not totally transparent to the user. To mount a physical file system into his or her private directory, the user first must know where the system is located. Once a system is mounted, however, the private hierarchy is used in a network-transparent way. For a new user who has not yet established a private hierarchy, the logical hierarchy is initialized to include a single "home" directory on some physical file system.

Second, the x-kernel file system accom-

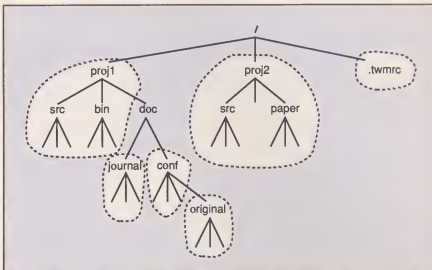


Figure 4. Private file hierarchy.

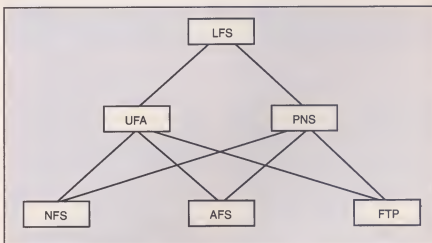


Figure 5. File protocol configuration.

modates multiple network file protocols for accessing each of the physical file systems mounted in a given logical file system. Each x-kernel file system spans a collection of heterogeneous file systems. For example, a given instance of the x-kernel file system might integrate one or more Sun network file systems, Andrew file systems, and so on. In addition, users can mount each others' private file systems into their own file system.

The central idea behind the logical file system is that it completely separates the directory function from the storage function. A private name space protocol called PNS implements the directory function, which translates the logical file name provided by the user into a handle for the file. A uniform file access protocol called UFA implements the storage function, which

accesses a file's data, given its handle. Both PNS and UFA depend on the underlying file access protocols. PNS uses the directory capabilities of those protocols (such as their Readir and Lookup operations), while UFA invokes their storage-related operations (such as their Read, Write, Get, and Put operations). All these protocols are configured into the kernel, as shown in Figure 5. Note that the logical file system, denoted LFS, is itself treated as a protocol in the kernel.

UFA translates logical file system operations into appropriate low-level protocol commands. It also manages a file cache. This is straightforward for file access protocols that closely match the Unix-like semantics of LFS; for example, the LFS Read operation maps to NFS' Read. UFA must do more work for file access proto-

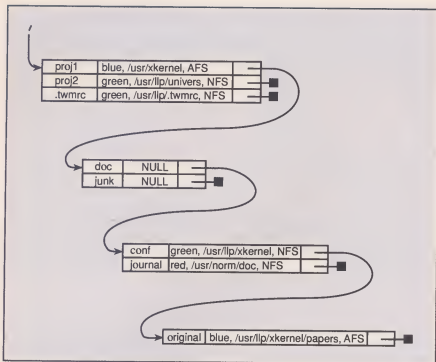


Figure 6. Skeleton directory structure.

cols that do not match LFS so closely. For example, FTP supports access to non-Unix file systems and uses whole-file transfer rather than block transfer. If a user opens a remote file, UFA uses FTP's Get operation to retrieve a copy of the file and caches it in a temporary file supported in one of the other available file systems. Subsequent Read and Write operations are translated into appropriate commands for the protocol that accesses the temporary file. When the file closes, FTP's Put command updates the remote copy of the file if the local copy was modified.

PNS implements a skeleton directory hierarchy. This directory structure keeps track of the boundaries between the underlying file systems. It is only a skeleton because the hierarchy within a given partition is maintained by the underlying file system, not by PNS. In other words, the logical hierarchy is superimposed over a collection of existing file hierarchies; much of the structure of the underlying hierarchies remains visible to the user.

Figure 6 shows a skeleton directory for the hierarchy in Figure 4. A directory entry has three basic elements: a portion of a path name in the logical hierarchy, a reference to a directory in some physical file system, and a pointer to another private directory. The second element is the most interesting. It identifies the host (server) that implements the desired file system, a directory

in that system, and the network protocol to access the system. In this example, red, blue, and green are three host names, where the NSF protocol accesses files on red and green, and AFS accesses files on blue.

To resolve a given path name, PNS traverses the longest possible path through the skeleton hierarchy. It then contacts the underlying file system to resolve the rest of the path name. For example, when resolving the name `/proj1/doc/conf/ieee/sec1`, PNS traverses the directory structure to the entry labeled `conf`, leaving `ieee/sec1` as the unresolved portion. PNS then invokes NFS directory operations on host green to resolve `ieee/sec1`. Because a user can mount another user's private name space in his or her own, PNS might need to contact another instance of itself to resolve a given name.

Although conceptually simple, this design has several subtle consequences. First, the underlying file systems are unaware they are participating in some user's private file system. Thus, when one physical file system is mounted under another, such as the subdirectory rooted at `original` in Figure 4, the former system contains no information pointing to the latter. All information needed to mount one directory under another is maintained at the workstation.

Second, a given private file system might contain directories that are not part

of any physical file system. For example, directory `/proj1/doc` in Figure 4 is not contained in any physical file system. Correspondingly, the entry for `doc` in Figure 6 does not identify any physical file system. The contents of directory `/proj1/doc` are found by following the pointer to the private directory that contains the entries `conf` and `journal`. This feature is a key reason why x-kernel uses the skeleton directory structure rather than a simpler prefix table, such as the one used in Sprite.

Third, path names in the private hierarchy could supersede names in the underlying file systems. For example, a directory named `/usr/xkernel/doc` on host blue would not be visible to the user because it would be hidden by the private directory `/proj1/doc`. That is, `/proj1/doc` replaces `/usr/xkernel/doc`. As another example, a file or directory named `/usr/xkernel/junk` on host blue would be hidden by the definition of `junk` in the private directory. Because `junk` is bound to a null physical file system and does not point to another private directory, its only purpose is to hide something in the underlying file system. Entries that hide files or directories in the underlying file system are not automatically displayed; they are only visible when using the special Lreaddir system call.

Finally, it is important to distinguish between file operations that modify the underlying file systems and operations that modify the logical file system. For example, invoking the operation `Mkdir (/proj2/tmp)` creates a new directory named `tmp` in the physical file system `/usr/lip/univers` on host green. Users invoke regular Unix file operations to affect the underlying file system; they must use the special `Lmkdir` operation to affect only the logical file system. For example, a user would have used `Lmkdir` to create directory `/proj1/doc`. The first argument to `Lmkdir` gives the logical directory name. The second argument identifies a physical directory to be mounted at the newly created directory. If the second argument is not given, then the directory exists only in the logical hierarchy, as is the case with `/proj1/doc`.

Command interpreter

Just as the file system provides uniform access to a heterogeneous collection of file systems, the x-kernel command interpreter provides uniform access to a heterogeneous collection of network services. The command interpreter differs from its

counterparts in other operating systems in that it invokes services located throughout the network, not just those services provided by the local kernel.

Consider the operation of the Unix shell. In Unix, the user types a command name, and the shell loads and executes the binary file that implements the command. For example, to compile a C program, the user enters the command `cc`. The shell then executes the file named `/bin/cc`. In other words, the command name is tightly bound to the resource — in this case a binary file — that implements the command. In contrast, the *x*-kernel command interpreter provides an interface to the entire network, so it must do more work to locate a suitable resource to implement a given command. To compile a C program, for example, the command interpreter also must locate an appropriate processor for the compilation.

Attribute-based names. The most novel aspect of the command interpreter is that it takes advantage of attribute-based names. An attribute-based name is a set of attributes or properties that describe a resource. A user looking for a fast processor on which to run a job might give an attribute-based name that includes the attributes "`mips>5`" and "`load<1.5`". A user looking for a printer might give an attribute-based name that includes the attributes "`resolution=300`" and "`location=723`".

An attribute-based name server is the mechanism that resolves attribute-based names. The *x*-kernel uses a server called *Univerts*.¹¹ A given *Univerts* name server maintains a database of attributes for each of a collection of resources. When given an attribute-based name, the server computes the set of resources described by the attributes. The server might maintain the following set of attributes for a processor:

```
(
  (name green)
  (address 192.12.69.22)
  (architecture 68020)
  (load 1)
  (mips 2)
  (owner "John Doe")
  (file_protocol (nfs rcp ftp))
  (exec_protocol (rpc rsh rexec))
)
```

When a name is submitted to *Univerts*, the server returns the entire set of attributes registered for each entry that matches the name. Thus, a user might submit the attributes "`mips>5`" and "`load<1.5`" to learn

the name and address. A user might also submit a name attribute to learn a processor's mips and load attributes. The server does not distinguish between "high level" names and "low level" names. Instead, users submit a collection of attributes they are interested in, and the name server returns all the attributes for the resources that match the given attributes.

Each name server also supports a programming (query) language that lets the user specify how the server should determine which resources match a given attribute-based name. For example, the query

```
(get_processor
 (architecture=68020 load<2))
```

reports all known attributes for those processors that have both 68020 architectures and a load less than 2. The query might return the set of resources

```
(
  (
    (name green)
    (address 192.12.69.22)
    (architecture 68020)
    (load 1)
    (mips 2)
    (owner John Doe)
    (file_protocol (nfs rcp ftp))
    (exec_protocol (rpc rsh rexec))
  )
  (
    (name blue)
    (address 192.12.69.33)
    (architecture 68020)
    (load 0)
    (mips 2)
    (owner John Smith)
    (file_protocol (nfs rcp ftp))
    (exec_protocol (rpc rsh rexec))
  )
)
```

As another example, a user interested only in learning the internet addresses of the processors returned by the first query might submit the following query to the name server:

```
(project
 (get_processor
  (architecture=68020 load<2))
 address)
```

Finally, a user who wants to learn about the owners of the processors that satisfy the original query might submit the following:

```
(dereference
 (get_processor
  (architecture=68020 load<2))
 owner)
```

perhaps yielding the result

```
(
  (
    (name "John Doe")
    (mailbox jd@arizona.edu)
    (login jd)
    (phone 555-1234)
    (office 723)
    (workstation green)
    (printer lw6)
  )
  (
    (name "John Smith")
    (mailbox js@arizona.edu)
    (login js)
    (phone 555-4321)
    (office 732)
    (workstation blue)
  )
)
```

Note that in addition to the desired set of attributes, each query consists of a composition of functions. In these examples, *project* and *dereference* are primitive functions provided by the name server (the former returns a particular attribute, and the latter returns all the objects referenced by a particular attribute), while *get_processor* is a user-defined function that knows about processors. In fact, *get_processor* is stored as an object in the name server's database.

Locating network services. The command interpreter uses attribute-based names to decouple command names from binary files to be executed on a particular host. Loosely speaking, invoking a command involves the following three steps:

- (1) The user gives a symbolic name for the command. The interpreter uses the logical file system to map the name to a command template, which includes a set of abstract attributes that partially describe the set of resources needed to provide the service.

- (2) The interpreter uses the abstract attributes to construct a query to *Univerts*, which returns concrete attributes describing resources that match the abstract attributes. The concrete attributes include information needed to access the resources, such as the host on which a given resource

```
(
-src array[-] of Filename:CopyIn "main.c";
-O;
-Def array[-] of string:
-Lib array[-] of Filename:CopyIn;
-obj Filename:CopyOut "a.out";
-host Hostname "architecture=68020,mips>2,load=min";
)
```

Figure 7. Example command template.

is implemented and the protocols needed to access it.

(3) Based on the concrete attributes, the interpreter invokes an access agent that establishes an appropriate environment, initiates the desired computation, and tears down the environment when the execution terminates.

The system defines an initial set of commands for most common services such as reading mail, printing documents, compiling source code, and text processing. In addition, users can define new commands by specifying their templates and implementing access agents for the commands.

Suppose a user wants to compile C programs on a Sun-3 workstation. The interpreter might accept a command line of the form

```
sun_cc -src /proj1/src/foo.c -Def De-
bug -obj /proj2/src/foo
```

The command interpreter must parse the command line, resolve the arguments, decide where the command should be executed, and arrange for its execution. The interpreter retrieves information needed to interpret the command line by reading a file `/commands/sun_cc` in the user's logical file system. This file contains two pieces of information: a template for the command and a pointer to an access agent for the command. The template describes the name and type of each argument to the command and how to establish an environment for executing the command.

In this example, `sun_cc` has the command template given in Figure 7. This template specifies that the command has six potential arguments. The keyword `Filename` indicates that the value of the argument must be resolved in the user's private file system. The keyword `Hostname` indicates that the value of the argument is an attribute-based name interpreted by a cer-

tain query that understands attributes for hosts (such as `get_processor`). The keywords `CopyIn` and `CopyOut` specify that the file (given as `Filename`) must be transferred into and out of the processor on which the command executes. The keyword `array[ ]` indicates that there may be multiple values for the same argument. In this template, the source files must be transferred to the execution processor before the command is executed, while the object file must be transferred back to the appropriate file system after the command is executed. Finally, the strings contained between double quotes (such as `"a.out"`) give the default value should the user not provide one. Arguments that specify no default value (such as `-O`) are ignored if not specified by the user.

Given this template, the command interpreter can parse the command line and resolve the arguments `/proj1/src/foo.c` and `/proj2/src/foo` in the user's logical file system. It also resolves the host name by invoking the query

```
(get_processor
 (architecture=68020 mips>2
 load=min))
```

Submitting this query to Univers might yield the answer

```
(
 (
 (name orange)
 (address 192.12.69.34)
 (architecture 68020)
 (load 1.1)
 (mips 3)
 (file_protocol (rcp ftp))
 (exec_protocol (rpc rsh rexec))
 (owner "John Smith")
 )
 )
```

Once the interpreter has received this result, it instantiates an access agent for the command. This agent selects an appropri-

ate file transfer protocol and remote execution protocol from those supported by the target host, transfers the source file to the host using the file transfer protocol, and invokes the service using the execution protocol. Given alternatives, the access agent uses a simple preference hierarchy to select a specific protocol: for example, RPC is preferred to RSH, NFS is preferred to FTP, and so on. Finally, the access agent uses the file protocol to copy the resulting object file to the specified file in the user's logical file system. Note that if a file transfer protocol like NFS is available, the access agent could set up the appropriate environment by simply establishing the correct working directory.

Commands can be defined to let the user explicitly give an attribute-based name for a resource. For example, the user could have used the `-host` option in the preceding example and submitted a different attribute-based name for a host. As another example, suppose the user wants to print an output file. In this case, the `Print` command can be defined to treat one of its arguments as an attribute-based name for a printer. The interpreter then resolves this name to learn the address of a specific printer. Thus, to output a file on John Doe's printer, the user types

```
print -P "owner=john doe" file
```

Similarly, the user might specify that the file be printed on a device that supports certain features. For example,

```
print -P "resolution=300,
font=palatino, location=Gould-
Simpson" file
```

By default — if the `-P` option is not specified — the `Print` command selects the nearest printer.

The x-kernel design is based on our belief that implementing communication protocols on a workstation is essential to accessing network resources. The more protocols the workstation supports, the more resources the user can access. Because many different protocols have been specified for various resources — for example, there are many file access protocols, remote procedure call protocols, directory service protocols, and so on — the workstation operating system must support the efficient execution of a wide variety of protocols. This is exactly what the x-kernel does.

All of the x-kernel's pieces have been implemented as of this writing, but some have not been integrated into the final, complete system. Specifically, the x-kernel itself and a large collection of protocols have been implemented and are now stable. We have implemented Sun RPC, Sprite RPC, Sun NFS, and FTP. We also plan to implement the client side of the Andrew file system, and we are now incorporating the X11 window system into the x-kernel. We plan to port the x-kernel to a RISC-based workstation soon. We will also enhance the kernel to dynamically load protocols; the current implementation statically loads all protocols configured into a given kernel. A public domain distribution of the x-kernel is available.

Prototypes of the file system and the command interpreter have been implemented on top of Unix; we are now incorporating them into the x-kernel. We do not expect this to be difficult, since both systems have been designed according to the "everything is a protocol" philosophy. The prototype file system implements a logical file system that depends on the FTP and NFS file-access protocols. The prototype command interpreter calls the Univers name server to resolve attribute-based names. A representative collection of command templates and access agents has also been implemented. We expect, however, that the exact syntax of the templates and the kind of information about resources that we store in the name server will evolve as we gain more experience with the interpreter. ■

Fastest Distributed Operating System," *Operating Systems Rev.*, Vol. 22, No. 4, Oct. 1988, pp. 25-34.

4. Sun Microsystems, *Network File System*, Mountain View, Calif., Feb. 1986.
5. J.H. Howard et al., "Scale and Performance in a Distributed File System," *ACM Trans. Computer Systems*, Vol. 6, No. 1, Feb. 1988, pp. 51-81.
6. Sun Microsystems, *Remote Procedure Call Programming Guide*, Mountain View, Calif., Feb. 1986.
7. N.C. Hutchinson et al., "RPC in the x-kernel: Evaluating New Design Techniques," *Proc. 12th ACM Symp. Operating System Principles*, Dec. 1989, pp. 91-101.
8. S.J. Leffler, W.N. Joy, and R.S. Fabry, "4.2 BSD Networking: Implementation Notes," *Unix Programmer's Manual*, Vol. 2, University of California at Berkeley, July 1983.
9. D.M. Ritchie, "A Stream Input-Output System," *AT&T Bell Laboratories Tech. J.*, Vol. 63, No. 8, Oct. 1984, pp. 311-324.
10. M. Accetta, "Mach: A New Kernel Foundation for Unix Development," *Proc. Summer Usenix*, July 1986, pp. 93-113.
11. M. Bowman, L.L. Peterson, and A. Yeatts, "Univers: An Attribute-Based Name Server," to appear in *Software — Practice and Experience*, John Wiley & Sons, New York, 1990.



Norman Hutchinson is an assistant professor in the Department of Computer Science at the University of Arizona. His research interests include programming languages and operating systems, specifically those intended for distributed environments.

Hutchinson received the BS degree in computer science from the University of Calgary in 1982 and the MS and PhD degrees in computer science from the University of Washington in 1985 and 1987, respectively. He is a member of the IEEE Computer Society and the ACM.



Sean O'Malley is a PhD candidate at the University of Arizona. His research interests are in networks, parallel processing, operating systems, and computer architecture. He is currently working on Avoca, a communications environment for constructing communications protocols.

O'Malley holds a BS degree in mathematical science from Stanford University and an MS degree in computer science from the University of Arizona. He is a member of the ACM.



Larry Peterson is an assistant professor in the Department of Computer Science at the University of Arizona. His research interests focus on communications in distributed systems, and he designed and implemented the Dragonmail conversation-based message system, the Profile naming service, and the Psync communication protocol. He is a member of the Internet working group on naming and is heading a NASA project to design and build an attribute-based naming service for the National Research Network.

Peterson holds a BS degree in computer science from Kearney State College and MS and PhD degrees in computer science from Purdue University. He is a member of the ACM.



Herman Rao is a PhD candidate at the University of Arizona. His research interests are in distributed systems, networks protocols, and user interfaces. He is currently working on Jade, a logical file system for a heterogeneous internet.

Rao holds a BS degree in mechanical engineering from the National Taiwan University and an MS degree in computer science from the University of Arizona.

Acknowledgments

This work is supported in part by National Science Foundation Grant CCR-8811423 and National Aeronautics and Space Administration Grant NCC-2-561.

References

1. D.R. Cheriton, "The V Distributed System," *Comm. ACM*, Vol. 31, No. 3, Mar. 1988, pp. 314-333.
2. J.K. Ousterhout et al., "The Sprite Network Operating System," *Computer*, Vol. 21, No. 2, Feb. 1988, pp. 23-36.
3. R. van Renesse, H. van Staveren, and S. Tannenbaum, "Performance of the World's

Readers can contact the authors at the Department of Computer Science, University of Arizona, Tucson, AZ 85721.

Scheduling Support for Concurrency and Parallelism in the Mach Operating System

David L. Black
Carnegie Mellon University

Significant changes in the use of multiprocessors require new support from operating system schedulers. Originally, multiprocessors increased throughput by running several applications at once, but no individual application ran faster. This use is giving way to parallel programming, which reduces the runtime of individual applications.

Parallel-programming models and languages often anticipate dedicated use of processors or an entire multiprocessor, but few machines are used in this fashion. Most modern general-purpose multiprocessors run a time-sharing operating system such as Unix. The shared-use model of these systems conflicts with the dedicated-use model of many programs, but the conflict is seldom resolved by restricting multiprocessor use to one application at a time.

Another impact on schedulers comes from the increased use of concurrency. The *application parallelism* for a multiprocessor application is the actual degree of parallel execution achieved, while *application concurrency* is the maximum degree of parallel execution that could be achieved with unlimited processors. For example, an application consisting of 10 independent processes running on a six-processor multiprocessor has an applica-

Traditional time-sharing schedulers are inadequate for parallel and concurrent programs, which require new techniques such as processor allocation and handoff scheduling.

tion parallelism of six, based on the six processors, and an application concurrency of 10, because it could use up to 10 processors. Application concurrency beyond hardware parallelism can improve hardware use; if one portion of the application blocks (for example, a disk or network operation), other portions can still proceed. The use of concurrency can also sim-

plify programming of concurrent applications by capturing the state of ongoing interactions in local variables of executing entities instead of in a state table.

Application parallelism and concurrency complicate two areas of scheduling: making effective use of the processing resources available to individual applications and dividing processing resources among competing applications. The problem of scheduling within applications seldom arises for serial applications because they can often be scheduled independently with little impact on overall performance. In contrast, the medium- to fine-grain interactions of parallel and concurrent programs may require scheduling portions of these programs together to achieve acceptable performance. Parallel applications that require a fixed number of processors complicate scheduling across applications. They make division of machine time more difficult and may introduce situations for which a fair-sharing policy is inappropriate. For example, an application configured for a fixed number of processors may be unable to cope efficiently with fewer processors.

At Carnegie Mellon University, developing the Mach operating system¹ for uniprocessors and multiprocessors produced some new approaches to scheduling. Mach provides flexible memory manage-

ment and sharing, multiple *threads* (control locus, program counter, registers) within a single address space or *task* for concurrency and parallelism, and a network-transparent communication subsystem. This communication subsystem is called the IPC (interprocess communication) subsystem for historical reasons; all communication in Mach is actually between tasks. The Mach kernel incorporates compatibility code derived from 4.3BSD Unix² that provides complete binary compatibility. Mach runs on a variety of uniprocessor and multiprocessor architectures, including the DEC VAX, Sun 3, IBM RP3, and Encore Multimax. Mach is available and supported as a product by a number of hardware vendors, including Next, Encore, and Omron. It is also the base technology for the OSF/1 operating system from the Open Software Foundation.

This article concentrates on the shared use of general-purpose uniprocessors and shared-memory multiprocessors, emphasizing support for common uniform-memory-access architectures that have all memory equidistant from all processors in terms of access time. This work is also applicable to nonuniform-memory-access machines, whose memory access times depend on the physical distance between the processor and the accessed memory, but it does not provide a complete solution to load-balancing problems for this class of machine.

Time-sharing scheduling

A major goal of time-sharing schedulers is allocating resources so that competing applications receive approximately equal portions of processor time. The "approximately equal" notion applies over periods of a few seconds to ensure interactive response in the presence of computation-bound jobs. In practice, this requires tracking processor usage and using the information in scheduling decisions. The simplest use, a decreasing-priority scheduler, continuously decreases a process' priority as it uses processor time and favors higher priority processes. Multics³ used such a scheduler and discovered its major disadvantage: on a heavily loaded system with many short-lived jobs, the priority of a lengthy job can decrease until little or no further processor time is available to it. The automatic priority depression of lengthy jobs in some versions

of Unix also exhibits this drawback.

To avoid the problem of permanently depressed priorities, it is necessary to elevate them in some manner. The two major elevation methodologies are event-based elevation and processor usage aging. Event-based elevation deliberately favors interactive response over computation-bound jobs. It elevates process priority through such events such as I/O completion. Elevations associated with events of interest must be determined in some manner, such as tuning under expected workloads to produce desired behavior. This methodology, used by VAX/VMS,⁴ assumes that jobs are either distinctly computation-bound or distinctly interactive and that interactive jobs are more important. Users whose interactive work consumes large amounts of processor time may not do well under this methodology. Such applications may not generate enough priority-raising events to offset the priority decreases caused by processor usage. Also, under this methodology, it may be necessary to retune priority elevations in response to workload or hardware changes.

Under the second priority-elevation methodology, processor usage aging, a scheduler elevates priorities by gradually forgetting about past processor usage, usually in an exponential fashion. For example, usage from one minute ago is half as costly as use within the past minute, usage from two minutes ago is half again as costly, and so on. As a result, the scheduler's measure of usage is an exponentially weighted average over the lifetime of a process. A simple exponential average is not desirable; it has the unexpected side effect of raising priorities when system load rises. This happens because, under higher loads, each process gets a smaller share of the processor, so its usage average drops, causing its priority to rise. These elevated priorities can degrade system response under heavy loads because no process accumulates enough usage to drop its priority. The 4.3BSD version of Unix solves this problem by making the aging rate depend on the load factor. Aging is slower in the presence of a higher load, which keeps priorities in approximately the same range.⁵ An alternative technique uses an overload factor to alter the rate at which usage accumulates. Under this technique, the usage accumulated by the scheduler is the actual usage multiplied by a factor based on the system load.

For a multiprocessor scheduler, the concept of fair sharing is strongly inter-

twined with that of load balancing. The goal of load balancing is to spread the system's computational load evenly among the available processors over time. For example, if three similar jobs are running on a two-processor system, each should average two-thirds of a processor. This often requires the scheduler to shuffle processes among processors to keep the load balanced. An important trade-off between load balancing and overhead is that optimal load balancing causes high scheduler overhead due to frequent context switches for load balancing. A uniprocessor time-sharing scheduler encounters similar issues in minimizing the number of context switches used to implement fair sharing.

Mach scheduler

The Mach operating system splits the usual process notion into task and thread abstractions, but the Mach time-sharing scheduler only schedules threads. The knowledge that two threads occur in the same task can be used to optimize the context switch between them but is not used to select which threads run. Favoring threads on this basis may not improve performance, depending on the hardware and application involved. It can also be detrimental to usage and load balancing because this favoritism may conflict with decisions needed to accomplish balancing.

Data structures. The primary data structure used by the Mach scheduler is the *run queue*, a priority queue of runnable threads implemented by an array of doubly linked queues. Mach uses 32 queues, so four priorities from the Unix range of 0 to 127 map to each queue. (More recent Mach kernels use a priority range of 0 to 31 so that queues and priorities correspond.) Lower priorities correspond to higher numbers and vice versa.

A hint is maintained that indicates the probable location of the highest priority thread. The highest priority thread cannot be at a higher priority than the hint, but it may be at a lower priority. This allows the search for the highest priority thread to start from the hint, potentially avoiding the search of a dozen or more queues because the highest possible priority for most user threads is 50 to match Unix.

Each run queue also contains a mutual exclusion lock and a count of threads currently enqueued. The count optimizes testing for emptiness by replacing a scan of

the individual queues with a comparison of the counter to zero. This eliminates the need to hold the run queue lock when the run queue is empty, thereby reducing potential contention for this important lock. The use of a single lock assumes that clock interrupts on the processors of a multiprocessor are not synchronized. Significant contention can be expected in the synchronized case; thus, a shared global run queue may not be appropriate.

Figure 1 shows a run queue with three threads. The queues containing the threads are doubly linked, but, for clarity, only the forward links are shown. The hint is 2, indicating that queues 0 and 1 are empty and can be skipped in a search for the highest priority thread.

When a new thread is needed for execution, each processor consults the appropriate run queues. The kernel maintains a *local run queue* for each processor and a shared *global run queue*. The local run queue is used by threads that have been temporarily bound to a specific processor for one of two reasons: (1) Although most of the Unix compatibility code in current Mach kernels has been parallelized,³ threads executing in the unparallelized portion are temporarily bound to a single, designated *master processor*; (2) interrupts that invoke Unix compatibility code are also restricted to this processor. The remaining use of the local run queues is by the processor allocation operation described in the section entitled "Processor allocation."

Mach is self-scheduling in that, instead of having threads assigned by a centralized dispatcher, individual processors consult the run queues when they need a new thread to run. A processor examines the local run queue first to give bound threads absolute preference over unbound threads. This precaution avoids bottlenecks by maximizing throughput of the unparallelized Unix compatibility code⁶ and improves processor allocation performance by preempting other operations. If the local queue is empty, the processor examines the global run queue. In either case, it dequeues and runs the highest-priority thread. If both queues are empty, the processor becomes idle.

Special mechanisms are used for idle processors. Most uniprocessor systems execute the idle loop by borrowing the kernel stack of the most recently run thread or process. On a multiprocessor this can be disastrous. If the thread resumes execution on another processor, the two processors will corrupt the thread's kernel stack.

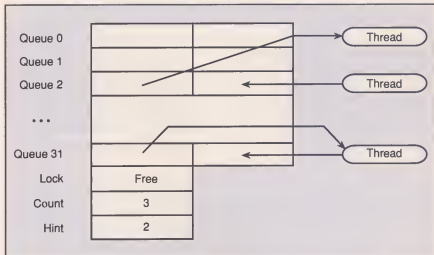


Figure 1. Mach run-queue structure.

To avoid stack corruption, Mach uses a dedicated *idle thread* for each processor. Idle processors are dispatched by a mechanism that bypasses the run queues. Threads that become runnable are matched with idle processors. The idle thread on each processor picks up the corresponding new thread and context switches directly to it, gaining performance by bypassing the run queues. This mechanism is one example of a general technique, called *handoff scheduling*, that context switches directly to a new thread without searching for it on a run queue.

Priority calculations. Thread priorities consist of a base priority plus an offset derived from recent processor usage. The base priority is set low enough to allow internal kernel threads, which perform critical kernel functions such as pageout, to run at higher priorities than user threads. The offset derived from usage is weighted by a load factor to preserve system responsiveness under load. If an adequate hardware source of time stamps exists, such as a 32-bit microsecond counter, the scheduler can be configured to base usage calculations on time stamps from this counter. This configuration eliminates the inaccuracies and distortions caused by statistical usage calculations (see Wendorf⁶ for details).

Mach uses the overload factor technique for processor usage aging. Aging overhead is distributed by making each thread responsible for aging its processor usage. Clock interrupts and events that unblock a thread cause it to check its local

copy of a counter against a global value that is incremented once a second. If these values differ, the thread ages its usage by a factor of 5/8 for each unit of difference. This results in each thread's accumulated usage being multiplied by 5/8 once a second. This figure was chosen for efficiency (two shifts and an add can implement multiplication by 5/8) and to produce behavior similar to other time-sharing systems. (The number used by 4.3BSD Unix depends on load; it varies from 1/2 to 2/3 for the 0.5 to 1.0 range.) The Mach scheduler uses a load factor of the number of runnable threads divided by the number of processors with a minimum factor of 1. This factor is calculated as an exponential average to smooth the impact of abrupt load changes.

The scheduling algorithm requires all threads to check their copy of the counter on a regular basis. The clock interrupt handler performs the check for running threads, and threads that are not running defer the check until they become runnable. Runnable threads with low priorities may spend long periods on a run queue while higher priority threads monopolize the processor. Such threads would run if they could check their copy of the counter and age their usage, but their low-priority positions prevent them from running to perform the check. To avoid such "starvation," an internal kernel thread scans the run queues every two seconds to age and elevate any threads caught in that situation. The two-second period, selected on the basis of experience with versions that exhibited starvation, produces acceptable behavior.

Managing context switches. The scheduler uses time quanta to minimize the number of context switches it causes. When a processor begins running a thread, that thread is assigned a time quantum. Periodic clock interrupts (every 10 to 100 milliseconds) decrement this quantum, and a rescheduling check occurs when the quantum expires if the thread is still running. This rescheduling check causes a context switch if there is another runnable thread of equal or higher priority; otherwise, the original thread continues running with a new quantum. The context switch mechanism uses an asynchronous system trap (AST) to avoid context switches in interrupt service routines. An AST is a general mechanism that allows an interrupt service routine to force the current thread to trap into the kernel from user mode. ASTs may be directly supported in hardware, implemented entirely in software, or a combination of the two. Priority reductions due to processor usage accumulated during a thread's first quantum do not cause context switches. This feature reduces the number of context switches for load and usage balancing by extending the time period between context switches.

Context switches also preempt lower priority threads when higher priority threads become runnable. On a uniprocessor, where an AST is requested as part of making the higher priority thread runnable, preemption occurs immediately. On a multiprocessor, if another processor must notice the presence of a higher priority thread on a run queue, preemption can take up to one clock-interrupt period. Mach doesn't use interprocessor interrupts for preemption; they haven't been necessary to achieve acceptable time-sharing behavior. Consequently, the scheduler doesn't maintain data structures that permit efficient identification of the lowest priority processor. However, as Mach evolves to support real-time applications, the addition of interprocessor interrupts for preemption is likely.

Programming models

Application programming models can introduce concurrency beyond hardware parallelism at two levels. An operating system can introduce concurrency by multiplexing Unix processes, Mach threads, or other independently schedulable entities (called *virtual processors*) onto hardware processors. A user-level library or language runtime can introduce

Table 1. Programming models for parallel and concurrent programming.

Relationship	Model
$MR = VP = PP$	Pure parallelism
$MR \geq VP = PP$	User concurrency
$MR = VP \geq PP$	System concurrency
$MR \geq VP \geq PP$	Dual concurrency

further concurrency by multiplexing language-level entities onto the VPs. Called *multiroutines*, these entities can be thought of as multiprocessor generalizations of coroutines. Special cases of multiroutines include coroutines (only one virtual processor) and the common programming notion of multiple threads (one virtual processor per multiroutine) found in many systems, including the Mach Cthreads library.⁷ Multiroutines and virtual processors can be identified in almost any parallel programming language implementation or environment. One example is an Ada runtime on Unix, which would use Unix processes as its virtual processors and Ada tasks as its multiroutines. The Mach Cthreads library uses Mach threads as its virtual processors and Cthreads as its multiroutines.

Parallel and concurrent programming models can be classified by the relationships among the multiroutines (MRs), virtual processors (VPs), and physical processors (PPs) they support (see Table 1). For *pure parallelism* models, the programmer's notion of concurrency is identical to the hardware parallelism. Compiler-generated code for parallel execution of loop bodies is a common example. *User concurrency* models introduce additional concurrency at the user level. An example would be programs based on a queue of user-defined tasks dequeued and executed in parallel by virtual processors. A *coroutine* model is a user concurrency model with exactly one physical processor and hence one virtual processor. *System concurrency* models, used by most multithreading packages and many parallel language runtime systems, introduce additional concurrency only at the system level. *Dual concurrency* models, a relatively new class, introduce concurrency at both the system and user levels. The programming model for an application depends on both the programming language or library and how it is used. For example, pure parallelism applications can be writ-

ten with a system concurrency library or language by creating only as many virtual processors as physical processors.

Fine-grain applications, which execute tens to hundreds of instructions between interactions with other multiroutines, cannot use system concurrency models. They require models with corresponding virtual and physical processors in which every virtual processor is always executing. This assumption is designed into the models' synchronization primitives, such as spin locks, and performance suffers if it is violated. Hence, programs using these models require dedicated physical processors. (Coroutine models, which use only one physical processor, are an exception.) The major disadvantage of these models is inefficient implementation of blocking system operations including page faults. Blocking a virtual processor in the operating system also blocks a physical processor, which wastes time when the physical processor is dedicated to the application.

The remaining model classes support blocking operations efficiently, but their potentially large synchronization overheads make them inappropriate for fine-grain applications. Their blocking operations are efficient because system concurrency can make an additional virtual processor available to use time relinquished by a blocked virtual processor. The blocking nature of synchronization in languages such as Ada and programming paradigms like message passing forces the use of models from these classes. The need for dedicated processors in these models is application dependent rather than inherent to the model. Applications for which parallel execution is important may need dedicated processors, while others may not. Communication or synchronization with nonexecuting virtual processors can be expensive because the operating system might not understand which processors are involved. Operating system support for such synchronization and communication can improve performance. But, even with this support, synchronization can consume hundreds of instructions, making these models inappropriate for fine-grain parallel applications.

Scheduling concurrency support

Applications using more virtual than physical processors can benefit from user input to scheduling decisions. Users may have application-specific information

about which virtual processors should or should not be running. The Mach scheduler is implemented in the operating system kernel but accepts user hints. These hints consist of local information involving the hint's thread, and possibly one other, so that users can avoid maintaining overall status information on their applications. The hints are based on two pieces of scheduling information that may be available when a thread attempts to communicate or synchronize with another thread that is not running. The first thread may be unable to make further progress until the communication or synchronization is complete, and it may know the identity of the thread that will complete the operation. For example, in a synchronization based on a message exchange, the initiating thread must block, and the identity of the thread that will reply to the message is often known.

The Mach scheduler accepts and uses two classes of hints: discouragement and handoff. A new primitive, `thread_switch`, allows simultaneous hints from both classes.

- *Discouragement hints*, which can be mild, strong, or absolute, indicate that the current thread should not run. Mild hints suggest giving up the processor to any other thread if possible. Strong hints go one step further and temporarily depress priority. Absolute hints block for a specified period.

- *Handoff hints* indicate that a specific thread should run instead of the current one.

Discouragement hints are useful for optimizing shared-memory synchronization in applications employing system concurrency. The lock holder's identity is not recorded by the common test-and-set instructions used to implement shared memory locks, making a handoff hint impossible. A mild discouragement hint yields the processor in the hope that the lock's holder will run. This can cause problems if more than one thread is yielding. They may yield to each other, with the result that no useful computation occurs. This situation can occur if the time-sharing scheduler gives the yielding threads higher usage-based priorities than the thread(s) they are waiting for. Absolute discouragement prevents this problem by blocking the threads, but the available time resolution based on clock interrupts is usually too coarse for medium- to fine-grain synchronization. Strong discouragement is a compromise that avoids the

Table 2. Synchronization experiment results (in microseconds).

Threads	3	4	5	6	7
Mild-ok	467 ± 3	633 ± 3	817 ± 6	973 ± 12	1,160 ± 19
Mild-bad	578 - 1,224	1,128 - 4,427	3.9 - 7.5 ms	8 - 607 ms	138 - 487 ms
Strong	897 ± 6	1,215 ± 9	1,434 ± 11	1,825 ± 10	2,130 ± 18
Handoff	413 ± 3	418 ± 5	417 ± 3	421 ± 4	428 ± 4

weaknesses of the other alternatives. Strong discouragement causes the scheduler to favor threads doing useful work, without the overhead of actually blocking those unable to proceed. A thread that issues a strong discouragement hint may explicitly cancel it when the desired event occurs; otherwise, the hint expires at a timeout supplied with the hint.

The second class of hint, handoff scheduling, directly "hands off" the processor to the specified thread, bypassing the internal scheduler mechanisms. Handoff scheduling may designate a thread within the same task or a different task on the same host to run next. A shared memory lock based on a compare-and-swap instruction can identify the target thread, or its identity may be available from the application's structure; for example, if a buffer is empty and only one thread fills it, then that thread should be run. One promising use of this technique addresses the "priority inversion" problem, where a low-priority thread holds a lock needed by a high-priority thread. The high-priority thread can detect this situation and hand the processor to the low-priority thread. When used from outside the kernel, handoff scheduling removes the specified thread from its run queue and runs it, avoiding a run queue search. Mach's message-passing subsystem also uses handoff scheduling inside the kernel to immediately run the recipient of a message. Use inside the kernel aids performance by avoiding the run queue entirely.

Performance. Two experiments — performed on an Encore Multimax with NS32332 processors having a speed of approximately 2 MIPS — have shown that scheduling hints benefit performance.

Synchronization hints. The first experiment investigated the use of hints for synchronizing with a thread that is not running. It used a multithreaded test program that synchronized with randomly selected threads. A shared variable contained a

thread number. That thread replaced it with another randomly chosen thread number, which then replaced it with another randomly chosen thread number, and so on. This program was restricted to a single processor, so it repeatedly synchronized with a nonexecuting thread. Threads not targeted for synchronization could use a scheduling hint to encourage the operating system to run the target.

Table 2 shows the elapsed time per synchronization in microseconds for different scheduling hints and numbers of threads as mean ± standard deviation. The mild discouragement hint exhibits two different behaviors. If all threads are at the same usage-based priority, the mild-ok line applies, and synchronization occurs quickly. If some of the threads are at different priorities, the mild-bad line applies, and the time per synchronization not only increases but exhibits pronounced variability. The mild-bad data is shown as ranges because the distributions are skewed toward higher frequencies at lower values. Multiple runs of the same test exhibit both behaviors unpredictably. This is strong evidence that, in many cases, mild discouragement is not an appropriate scheduling hint. Strong discouragement is an appropriate alternative. Its behavior is far more stable, and its performance is better than most bad cases of mild discouragement. The synchronization times from the five-thread test cases for all hints is slightly better than expected, based on the other results. This is probably because the number of threads is a divisor of the 10-Hertz clock frequency that drives the scheduler, so stable behavior is more likely. Absolute discouragement was not tested because it would result in times on the order of 100 milliseconds per synchronization, given the hardware clock-interrupt frequency of 10 Hertz.

The results show the benefits of scheduling hints. Running the program without scheduling hints yields times of one-half to one second per synchronization, demonstrating the potential performance pen-

Table 3. Message-passing handoff results (in microseconds).

Messages	Remote Procedure Call		One-way	
	No	Yes	No	Yes
No Handoff	1,914 $\pm$ 11	1,630 $\pm$ 6	857 $\pm$ 5	432 $\pm$ 3
Handoff	1,848 $\pm$ 8	1,628 $\pm$ 7	861 $\pm$ 10	429 $\pm$ 4

alties for ignoring the problem. If the threads are at the same priority, context switching is effective; however, if that is not the case, priority inversions cause poor results. Strong discouragement performs predictably but is slower than the best cases of mild discouragement because of the timeout costs associated with priority depressions. These costs also account for the increasing difference between the strong and mild-ok results as the number of threads increases. Handoff scheduling performs the best and is significantly faster than sending a message, since no time is spent formatting and transporting the message or blocking to wait for it. These results suggest that system-concurrent applications require strong-discouragement support and that handoff scheduling is effective if the required information is available. These are worst-case results, but they do indicate the relative performance of the hints.

IPC handoff scheduling. The second experiment concerned the performance benefits of using handoff scheduling in the kernel. It used a message-passing exerciser to measure the performance impact of handoff scheduling in Mach's network-transparent communication subsystem, the IPC. The experiment involved exchanging messages between two threads in a task on both single and multiple processor configurations. The key difference between the uniprocessor and multiprocessor experiments is the availability of an idle processor to run the recipient thread. Hence, the uniprocessor results are also applicable to multiprocessors when no idle processors are available. The experiment was run for both unidirectional and bidirectional (remote procedure call (RPC)) message exchanges.

Table 3 shows the results in microseconds of elapsed time per exchange as mean $\pm$ standard deviation. The time differences are statistically significant only for RPCs without idle processors. The RPC case with idle processors benefits from a hand-

off in the dispatching code (see "Mach Scheduler, Data Structures," above). This handoff was not disabled for these experiments. It is not specific to the IPC system, and disabling it requires scheduler modifications that impact the critical-context switch path. One-way message exchanges do not gain performance from handoff scheduling for two reasons. First, in the absence of handoff scheduling, a sender can queue multiple messages before context switching to the receiver. Second, on a multiprocessor, the sender and receiver can run in parallel with complete overlap.

Based on these results, Mach is configured to use handoff scheduling for RPC when no idle processor is available. Current Mach kernels can hand off only once per RPC because the RPC send half is implemented separately. Thus, the sender hands off to the receiver before the sender is queued for the reply. When the reply comes back, no thread is queued and no handoff takes place. The Mach IPC system is being redesigned to incorporate a bidirectional message primitive that can hand off in both directions. Similar functionality exists in other systems, such as the Topaz operating system developed at DEC's Systems Research Center for the Firefly.⁸

Alternatively, scheduling hints could be combined into higher level kernel synchronization primitives, such as semaphores or condition variables. Higher level primitives can provide a cleaner interface, by hiding more scheduler details, and can simplify the implementation of a library or language runtime that uses them. The disadvantages are that languages and libraries must use these primitives to influence the scheduler, and the primitives may be specialized toward some languages or classes of applications. This specialization can impact performance if the primitives are not a good match to the programming language or model. The Topaz system uses this approach, but specialization is not an issue in that environment because most programming is done in Modula-2+.⁸

Processor allocation

Gang scheduling, which guarantees simultaneous scheduling of an application's components, is one use for processor allocation in multiprocessor operating systems. It is necessary for fine-grain parallel applications whose performance severely degrades when any part of the application is not running, but it is also applicable to other classes of parallel programs. The need for gang scheduling is widely recognized, and implementations exist on a variety of multiprocessors. This section describes the design and implementation of Mach's processor allocation facility.

Design. Because Mach supports a multitude of applications, languages, and programming models on a variety of multiprocessor architectures, flexibility is the driving factor in the design of its processor allocation facility. This flexibility has several aspects.

- The facility should be capable of allocating processors to applications written in different languages with different programming models. Binding threads to individual processors is not sufficient because applications that use system concurrency need to bind a pool of threads to a pool of processors. Such *pool binding* improves performance by allowing any thread to run on a processor vacated by a blocked thread.

- The facility should be adaptable to the different multiprocessor architectures that can run Mach. In particular, it should support uniform-memory access (UMA) and nonuniform-memory-access (NUMA) architectures without major changes to the kernel interface.

- The facility should accommodate different policies — sure to exist at various installations — concerning who can allocate how many processors, when, and for how long. Changes to these policies should not require rebuilding the kernel.

- Finally, the facility should offer applications complete control over which threads execute on which processors, but it should not force implementation on applications not wanting this degree of control.

Mach's processor allocation facility meets these goals by dividing the responsibility for processor allocation among the three components shown in Figure 2:

- kernel, performs allocation mechanisms only;

- server, implements allocation policy;
- application, requests processors from server and uses them. Can control their use if desired.

The server must be privileged, to gain direct control over processors. As the component most affected by changes in usage policies or hardware configuration, the server is designed to be much easier to replace or reconfigure than the kernel. The design assumes that processors will be dedicated to applications for seconds or longer, rather than milliseconds, to amortize the overhead of crossing boundaries among components. The application-to-server interface is not specified because it will be affected by changes in usage policy and hardware architecture. Some servers may require applications to provide authentication information to establish their right to use certain processors, while other servers may require information describing the location of requested processors in a NUMA architecture. The kernel interface does not change from machine to machine, but some calls return machine-dependent information.

The allocation facility adds two new objects to the Mach kernel interface, the *processor* and the *processor set*. *Processor* objects correspond to and manipulate physical processors. *Processor set* objects are independent entities to which threads and processors can be assigned.

Processors only execute threads assigned to the same processor set and vice-versa, and every processor and thread is always assigned to a processor set. If a processor set has no assigned processors, then threads assigned to it are suspended. Assignments are initialized by an inheritance mechanism. Each task is also assigned to a processor set, but this assignment is used only to initialize the assignment of threads created in that task. In turn, each task inherits its initial assignment from its parent upon creation, and the first task in the system is initially assigned to the *default processor set*. Thus, in the absence of explicit assignments, every thread and task in the system inherits the first task's assignment to the default processor set. All processors are initially assigned to the default processor set, and at least one processor must always be assigned to it so that internal kernel threads and important daemons can be run.

Processors and processor sets are represented by Mach ports. Because access to a port requires a kernel-managed capability for that port, or *port right*, entities other

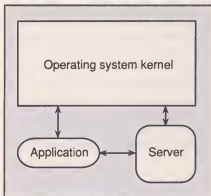


Figure 2. Processor allocation components.

than the appropriate server and/or application lack the required port right(s) and cannot interfere with allocations. Processor sets also have a name port for identification and status queries, but this port cannot be used to manipulate the processor set.

Responsibility for the allocation and use of dedicated processors is divided among the application, server, and kernel. The application controls the assignment of tasks and threads to processor sets. The server controls the assignment of processors to processor sets. The kernel does whatever the application and server ask it to do.

Here's how an application might allocate six processors for its use:

- (1) **Application** $\Rightarrow$ **Kernel** Create processor set.
- (2) **Application** $\Rightarrow$ **Server** Request six processors for processor set.
- (3) **Application** $\Rightarrow$ **Kernel** Assign threads to processor set.
- (4) **Server** $\Rightarrow$ **Kernel** Assign processors to processor set.
- (5) **Application** Use processors.
- (6) **Application** $\Rightarrow$ **Server** Finished with processors (optional).
- (7) **Server** $\Rightarrow$ **Kernel** Reassign processors.

This example illustrates three important features of the allocation facility.

- The application creates the processor set and uses it as the basis of its communication with the server, freeing the server from dependence on the internal structure of the application.

- Only one processor set is used. The scheduling algorithms, described earlier, function within each processor set, and if the task in this example contains six or fewer threads, there will be no context

switches to shuffle the threads among the allocated processors.

- The server does not need the application's cooperation to remove processors from it. The server retains complete control over the processors at all times because it retains the access rights to the processor objects. Removing processors without the application's cooperation should not be necessary for well-behaved applications, but it can be useful for a runaway application that has exceeded its allocated time.

This design meets its flexibility goals. It supports different programming models and languages; it can assign processors to individual processor sets or to one or more common sets that match application requirements. Assigning one processor to each of several processor sets gives the application complete control over which threads run on which processors.

Isolating scheduling policy in a server simplifies changes for different hardware architectures and site-specific usage policies. NUMA machines can use this allocation facility to match processor sets to clusters of processors with identical memory access characteristics. This disables the kernel scheduler's load balancing between clusters, which is a minimum requirement for scheduling on NUMA machines. The facility does not replace the disabled load balancing, but, by design, the kernel interface makes sufficient information available for a user-mode implementation of a NUMA load balancer.

Implementation. Kernel implementation of processor sets extends the Mach time-sharing scheduler. The data structure for each processor set contains a run queue that is used as the global run queue for processors assigned to the set. A list of idle processors is also maintained on a per-processor-set basis because a processor can only run threads assigned to its current set. The processor-set data structure also heads individual linked lists that are threaded through the data structures of assigned tasks, threads, and processors so that these entities can be found and reassigned when the processor set is terminated. In addition, the data structure contains state information required to run the time-sharing scheduling algorithm, the identities of ports that represent the set, and a mutual-exclusion lock to control access to the data structure. Thread assignment suspends the thread involved so that it can remain suspended if assigned to a

processor set with no processors. Inter-processor interrupts are used for thread and processor assignment as needed.

Table 4 shows times required by basic operations in the processor allocation system as mean $\pm$ standard deviation in microseconds. The "self" and "other" cases of thread assignment correspond to a thread assigning itself and a thread assigning another thread.

Special techniques manage processor-to-processor set relationships. Code in a critical scheduler path reads a processor's assignment as part of finding a thread to run on that processor. To optimize this common case against infrequent assignment changes, each processor can change only its own assignment. This restriction avoids the need for a mutual-exclusion lock because a processor cannot look for a new thread and change its assignment simultaneously. The cost to the assignment operation is that it must temporarily bind a thread to the processor while changing the assignment. An internal kernel thread, the *action thread*, serves this purpose. Current kernels use only one action thread but are designed to accommodate more. The processor assignment interface lets a server avoid synchronizing with each assignment's completion, so a server thread can exercise the parallelism available from multiple action threads.

Gang-scheduling server. To demonstrate the utility of this work, we implemented a simple processor-allocation server for gang scheduling. The server is a batch scheduler for processors that schedules requests on a greedy first-come, first-served basis, subject to the number of available processors. The server is configured for a maximum allocation of 75 percent of the machine for, at most, 15 minutes at a time.

The server implementation uses two threads: one to manage processors and another to communicate with applications. The primary interaction between the threads is via operations on shared data structures describing the requests, but the interaction thread sends a message to the processor thread when an immediate change in processor assignment is needed. One such situation is the receipt of an allocation request that can be satisfied immediately.

Library routines are available to hide the server interfaces, so an application can make a single call indicating how many processors it wants for how many seconds. This routine contacts the server, arranges

Table 4. Allocation operation performance.

Operation	Time (μ s)
Create processor set	2,250 $\pm$ 50
Assign processor	4,772 $\pm$ 28
Assign thread (self)	1,558 $\pm$ 25
Assign thread (other)	2,624 $\pm$ 185

the allocation, and returns when the server has begun assigning the requested processors. The routine takes about 35 milliseconds to allocate one processor plus about 5 milliseconds for additional processor. This overhead is acceptable, given expected allocations of tens of seconds to tens of minutes.

At Carnegie Mellon University, researchers and students in an undergraduate parallel programming course have successfully used this server and library interface in measuring the performance of parallel programs. The server removed most of the usual administrative obstacles to obtaining dedicated machine time. In addition, it demonstrated the utility of implementing policy in a separate server; server crashes did not crash the operating system.

Many extensions and changes to the policy implemented in the *cpu_server* are possible. Since it is a batch scheduler, techniques originally developed for batch scheduling of memory, such as assigning higher priority to shorter requests, are applicable. In addition, the server could be extended to allow some users higher or absolute priority in allocating processors or to allow allocation of more processors during light usage periods. Finally, the server could be entirely replaced by a server that implements a different scheduling policy. One promising new policy is to vary the number of processors available to applications according to overall demand. A server with this policy would tell applications to reconfigure when it changes the number of processors available. Researchers at Stanford are pursuing this approach and have implemented such a server under Mach with good initial results.⁹

Related work

Previous work^{10,11} on *policy mechanism separation* proposed separating the

scheduler into two pieces, with mechanisms implemented in the operating system and policy decisions made by a user-mode *policy module*. This work, which considered only the problem of scheduling within applications, encountered two problems.

The first problem was the overhead of crossing the boundary between the operating system and an application to access a policy module. Crossing the boundary required operating-system implementation of short-term policy, which, to the detriment of the policy modules, made it more efficient to delay long-term policy decisions.

The second difficulty, revealed through experience with the resulting systems, was that most applications did not use the available flexibility. One reason for this lies in the inherent complexity of policy modules; most nontrivial instances require an intricate scheduler implementation.^{10,11}

Our use of policy/mechanism separation avoids both problems. Processor allocation decisions are infrequent enough to effectively amortize boundary crossing costs, and the complex policy implementation resides in a server that is implemented once for a system rather than in a module that must be customized to each application.

Another body of related work concerns coscheduling, a multiprocessor scheduling policy that attempts simultaneous scheduling of an application's components but makes no guarantee of success. This policy was originally proposed for medium-grain, parallel message-passing applications with hundreds to thousands of instructions between interactions. Such applications benefit from coscheduling but achieve reasonable performance without it.

The major work on coscheduling was done for the Medusa operating system on Cm*.¹¹ It is not directly applicable to current multiprocessors because the techniques depend on synchronized clocks and a memory structure that precludes short-term load balancing. In contrast, the uniform shared-memory machines that are our primary interest do need short-term load balancing and do not have synchronized clocks.

The Alliant Concentrix scheduler, described by Jacobs,¹² is an alternative approach to processor allocation and control. This scheduler supports a fixed number of scheduling classes and uses a *scheduling vector* for each processor to indicate

which classes should be searched for work in what order. Each processor cycles through a set of scheduling vectors based on time durations associated with each vector, typically fractions of a second. Processes are assigned to scheduling classes by their characteristics or a system call available to privileged users and applications. This scheduler is oriented toward dividing processors among statically defined classes of applications over short periods of time. This contrasts with Mach's orientation of dedicating processors to applications over longer periods of time. Mach's processor sets can be created dynamically in contrast to Concentrix's fixed number of scheduling classes. Scheduling servers could be implemented by reserving some scheduling classes for their exclusive use, but the static class and vector definitions appear to restrict the flexibility available in forming processor sets. The Concentrix scheduler also enforces a more restrictive version of gang scheduling in which a blocking operation by any thread blocks the entire gang. This limits it to applications not using system concurrency and makes parallel handling of blocking operations, such as I/O and page faults, all but impossible.

Many parallel and concurrent applications cannot be scheduled acceptably by traditional time-sharing means. Dedicated processors are required to obtain acceptable performance from some parallel applications. For concurrent applications, communication and synchronization performance can be improved by taking advantage of application-specific scheduling information. Mach's scheduler has been enhanced to meet these challenges.

Mach allows concurrent programs to provide handoff and discouragement hints to influence scheduling decisions. Of these two, handoff hints are more effective; naming the next thread to run bypasses much of the scheduler logic that normally makes this decision. These hints are based on local information that is easy to obtain and provide to the scheduler. Using local information and hints avoids the overhead and complexity drawbacks of previous work based on application-specific policy modules.

Obtaining and scheduling dedicated processors is supported by Mach's processor allocation and control facilities. The Mach kernel implements only allocation mechanisms; policy decisions are made

by a privileged server that is much easier to reconfigure or replace than the kernel. This provides a wide degree of flexibility to implement allocation policies and accommodate different multiprocessor architectures. This design also accommodates applications based on different programming models.

Code to implement the Mach features described in this article will be available from several sources. Future Mach releases from Carnegie Mellon University and Mt. Xinu are expected to support these features, but the current 2.5 and 2.6 MSD releases do not. These features are a part of the OSF/1 operating system from the Open Software Foundation and are or will be available in some vendor versions of Mach (for example, Encore's Mach for the Multimax). ■

Acknowledgments

The initial design of the run queue structure is due to Avadis Tevanian, Jr., and is similar to structures used in various versions of Unix. The initial implementation of handoff scheduling for kernel use was done by Richard Rashid. The author would like to thank all members of the Mach project for making this research possible and *Computer's* referees and editors for suggestions that greatly improved the structure and presentation of this article.

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), monitored by the Space and Naval Warfare Systems Command under Contract N00039-87-C-0251. However, the views and conclusions contained in this article are those of the author and should not be interpreted as representing official policies, either expressed or implied, of DARPA or the US government.

Winter 1990 *Unix Conf.*, Jan. 1990, pp. 297-312.

6. J.W. Wendorf, "Operating System/Application Concurrency in Tightly Coupled Multiple-Processor Systems," PhD thesis, Tech. Rep. CMU-CS-88-117, Carnegie Mellon University, Computer Science Dept., Pittsburgh, 1987.
7. E.C. Cooper and R.P. Draves, "C Threads," Tech. Rep. CMU-CS-88-154, Carnegie Mellon University, Computer Science Dept., Pittsburgh, 1988.
8. C.P. Thacker, L.C. Stewart, and J.E.H. Satterthwaite, "Firefly, A Multiprocessor Workstation," *IEEE Trans. Computers*, Vol. 37, No. 8, Aug. 1988, pp. 909-920.
9. A. Tucker and A. Gupta, "Process Control and Scheduling Issues for Multiprogrammed Shared Memory Multiprocessors," *Proc. 12th ACM Symp. Operating Systems Principles*, Dec. 1989, pp. 159-166.
10. W.A. Wulf, R. Levin, and S.P. Harbison, *Hydra/C.mmp: An Experimental Computer System*, McGraw-Hill, New York, 1981.
11. E.F. Gehring, D.P. Siewiorek, and Z. Segall, *Parallel Processing: The Cm* Experience*, Digital Press, Maynard, Mass., 1987.
12. H. Jacobs, "A User-Tunable Multiple Processor Scheduler," *Proc. Winter 1986 Unix Conf.*, Jan. 1986, pp. 183-191.



References

1. R.F. Rashid, "Threads of a New System," *Unix Review*, Vol. 4, Aug. 1986, pp. 37-49.
2. S.J. Leffler et al., *Design and Implementation of the 4.3BSD Unix Operating System*, Addison-Wesley, Reading, Mass., 1989.
3. E.I. Organick, *The Multics System: An Examination of Its Structure*, MIT Press, Cambridge, Mass., 1972.
4. L.J. Kenah and S.F. Bate, *VAX/VMS Internals and Data Structures*, Digital Press, Maynard, Mass., 1984.
5. A. Langerman et al., "A Highly-Parallelized Mach-based Vnode Filesystem," *Proc.*

David L. Black is a graduate student and PhD candidate in the School of Computer Science at Carnegie Mellon University, where he has worked on the design and implementation of the Mach operating system since 1986. His research interests are in operating systems, with emphasis on multiprocessor and scheduling issues.

Black received BA and MA degrees in mathematics and the BSE degree in computer science and engineering from the University of Pennsylvania in 1983. He received the MS degree in computer science from Carnegie Mellon University in 1985. He is a member of the ACM, IEEE, the Computer Society, and several honorary societies.

Readers may contact Black at Carnegie Mellon University, School of Computer Science, Pittsburgh, PA 15213 or, by e-mail, at david.black@cs.cmu.edu.

Amoeba

A Distributed Operating System for the 1990s

Sape J. Mullender and Guido van Rossum
Centre for Mathematics and Computer Science

Andrew S. Tanenbaum, Robbert van Renesse, and
Hans van Staveren
Free University of Amsterdam

In the next decade, computer prices will drop so low that 10, 20, or perhaps 100 powerful microprocessors *per user* will be feasible. All this computing power will have to be organized in a simple, efficient, and fault-tolerant system that is easy to use. The basic problem with current networks of PCs and workstations is that they are not transparent; that is, users are aware of the other machines. The user logs into one machine and uses that machine only, until doing a remote login to another machine. Few if any programs take advantage of multiple CPUs, even when all are idle.

We envision a system for the 1990s that will appear to users as a single, 1970s centralized time-sharing system. Users will not know which processors their jobs are using (or even how many), where their files are stored (or how many replicated copies are maintained to provide high availability), or how processes and machines are communicating. All resources will be managed completely and automatically by a distributed operating system.

Few such systems have been designed,

**The Amoeba
distributed operating
system appears to
users as a centralized
system, but it has the
speed, fault tolerance,
security safeguards,
and flexibility
required for the 1990s.**

and even fewer have been implemented. Fewer still are actually used by anyone yet. An early distributed system was the Cambridge system.¹ Later systems were Locust,² Mach,³ the V-Kernel,⁴ and Chorus.⁵ Here we describe Amoeba, a distributed

system developed at the Free University and the Centre for Mathematics and Computer Science in Amsterdam. Amoeba combines high availability, parallelism, and scalability with simplicity and high performance.

Although distributed systems are necessarily more complicated than centralized systems and tend to be much slower, we have worked hard to achieve extremely high performance: Amoeba is already one of the fastest distributed systems (on its class of hardware) reported so far, and future versions will be even faster. With the current implementation, a remote procedure call can be performed in 1.4 ms on Sun-3/50 class machines. The file server can deliver data continuously at 677 Kbytes per second.

The Amoeba software is based on objects. An object is a piece of data on which well-defined operations can be performed by authorized users, independent of the user's and object's locations. Objects are managed by server processes and named using capabilities chosen randomly from a sparse name space.

A process is a segmented address space shared by one or more threads of control. Processes can be created, managed, and debugged remotely. Operations on objects are implemented using remote procedure calls.

Amoeba has a unique, fast file system split into two parts: The bullet service stores immutable files contiguously on the disk; the directory service gives capabilities symbolic names and handles replication and atomicity, eliminating the need for a separate transaction management system.

To bridge the gap with existing systems, Amoeba has a Unix emulation facility consisting of a library of Unix system call routines that make calls to the various Amoeba server processes.

Most classical distributed systems literature describes work on parts of or aspects of distributed systems: distributed file servers, distributed name servers, distributed transaction systems, and so on. Here we discuss the whole system, covering most of the traditional operating system design issues, including communication, protection, the file system, and process management. We explain not only what we did but also why we did it.

Overview of Amoeba

The Amoeba project⁶ has been under way for nearly 10 years and has seen numerous system redesigns and reimplementations as design flaws became glaringly apparent. This article describes the Amoeba 4.0 system, released in 1990.

Hardware architecture. As Figure 1 shows, the Amoeba hardware consists of four components: workstations, pool processors, specialized servers, and gateways. The workstations execute only processes that require intense user interaction—for example, window managers, command interpreters, editors, and CAD/CAM graphical front ends. Most applications, however, do not interact much with the user and are run elsewhere.

Amoeba's processor pool provides most of the computing power. Typically it consists of many single-board computers, each with several megabytes of private memory and a network interface. The Free University, for example, has 48 such machines. A pile of diskless, terminalless workstations can also be used as a processor pool.

When a user has an application to run—for example, building a program consist-

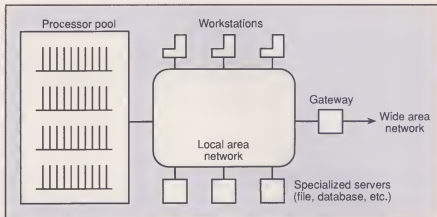


Figure 1. Four components of the Amoeba architecture.

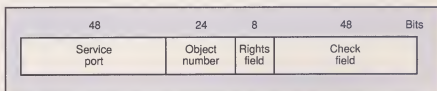


Figure 2. The structure of a capability. The service port identifies the service that manages the object. The object number specifies the object (for example, which file). The rights field determines which operations are permitted. The check field provides cryptographic protection to keep users from tampering with the other fields.

ing of dozens of source files—a number of processors can be allocated to run many compilations in parallel. When the user is finished, the processors are returned to the pool for other work. Although the pool processors are all multiprogrammed, the best performance is obtained by giving each process its own processor, until the supply runs out.

The processor pool allows us to build a system in which the number of processors exceeds the number of users by an order of magnitude or more, something quite impossible in the personal workstation model of the 1980s. The software has been designed to treat the number of processors dynamically, so processors can be added as the user population grows. When a few processors crash, some jobs may have to be restarted and the computing capacity is temporarily lowered, but otherwise the system continues normally, providing a degree of fault tolerance.

Specialized servers, the third system component, are machines for running dedicated processes with unusual resource demands. For example, it is best to run file servers on machines that have disks.

Finally, there are gateways to other Amoeba systems that can be accessed only over wide area networks. For a project sponsored by the European Community we built a distributed Amoeba system that spanned several countries. The gateway protects local machines from the idiosyncrasies of protocols that must be used over the wide area links.

Why did we choose this architecture instead of the traditional workstation model? As it becomes possible to give each user 10 to 100 processors, centralizing the computing power will allow incremental growth, fault tolerance, and the ability for a large job to obtain a large amount of computing power temporarily. Current systems have file servers, so why not let them have computer servers as well?

Amoeba software architecture. Amoeba is an object-based system using clients and servers. Client processes use remote procedure calls to send requests to server processes for carrying out operations on objects. Each object is both identified and protected by a *capability*, as Figure 2 shows. Capabilities have the set

Amoeba Interface Language

Interfaces for object manipulation are specified in a notation called the Amoeba Interface Language.¹ AIL resembles the notation for procedure headers in C, but it has some extra syntax for automatic generation of client and server stubs. The Amoeba class for standard manipulations on filelike objects, for instance, could be specified as follows:

```
class basic_io [1000..1199] {  
    const    BIO_SIZE = 30000;  
  
    bio_read(*,  
             in unsigned offset,  
             in out unsigned bytes,  
             out char buffer[bytes:bytes]);  
  
    bio_write(*,  
             in unsigned offset,  
             in out unsigned bytes,  
             in char buffer[bytes:BIO_SIZE]);  
};
```

The names of the operations, `bio_read` and `bio_write`, must be globally unique. They conventionally start with an abbreviation of the name of their class. The first parameter, indicated by an asterisk, is always a capability of the object to which the operation refers. The other parameters are labeled "in," "out," or "in out" to indicate whether they are input or output parameters to the operation, or both. Specifying this allows the stub compiler to generate code to transport parameters in only one direction.

The number of elements in an array parameter can be specified by `[n:m]`, where *n* is the actual number of elements

in the array and *m* is the maximum number. In an out array parameter such as `buffer` in `bio_read`, the maximum size is provided by the caller. In `bio_read`, it is the value of the in-parameter bytes. The actual size of an out array parameter is given by the callee and must be less than the maximum. In `bio_read` it is the value of the out parameter bytes — the actual number of bytes read. On an in array parameter, the maximum size is set by the interface designer and must be a constant, while the actual size is given by the caller. In `bio_write`, it is the in value of bytes.

This AIL specification tells the stub compiler that the operation codes for `basic_io` must be allocated in the range 1000 to 1199. A clash of operation codes for two different classes matters only if these classes are both inherited by another, bringing them together in one interface. Currently, each group of people designing interfaces has a different range from which to allocate operation codes. Later we hope to allocate operation codes automatically.

The AIL stub compiler can generate client and server stub routines for a number of programming languages and machine architectures. For each parameter type, marshalling code is compiled into the stubs that convert data types of the language to AIL data types and internal representations. Currently, AIL handles only fairly simple data types (Boolean, integer, floating point, character, string) and records or arrays of them. However, it can easily be extended with more data types when the need arises.

Reference

1. G. van Rossum, "AIL — A Class-Oriented Stub Generator for Amoeba," *Proc. Workshop on Experience with Distributed Systems*, Springer-Verlag, Berlin, to be published in 1990.

of operations that the holder may carry out on the object coded into them, and they contain enough redundancy and cryptographic protection to make guessing an object's capability infeasible. Keeping capabilities secret by embedding them in a huge address space is the key to protection in Amoeba. Because of the cryptographic protection, capabilities can be managed outside the kernel, by user processes themselves.

Objects are implemented by the server processes that manage them. Capabilities have the identity of the object's server encoded into them (the service port) so that, given a capability, the system can easily find a server process that manages the corresponding object. The remote procedure call system guarantees that requests and replies are delivered only once, and only to authorized processes.

Although at the system level objects are identified by their (binary) capabilities, at

the level where most people program and work, objects are named using a symbolic hierarchical naming scheme. The directory service maintains a mapping of ASCII path names onto capabilities and has mechanisms for performing atomic operations on arbitrary collections of name-to-capability mappings.

Amoeba has already gone through several generations of file systems. Currently, one file server is used almost to the exclusion of all others. The bullet service (which got its name from being faster than a speeding bullet) is a simple file server that stores immutable files as contiguous byte strings both on disk and in its cache.

The Amoeba kernel manages memory segments, supports processes containing multiple threads, and handles interprocess communication. The process management facilities allow remote process creation, debugging, checkpointing, and migration, all using a few simple mechanisms ex-

plained in a later section.

All other services (such as the directory service) are provided by user-level processes, in contrast to, say, Unix, which has a large monolithic kernel for these services. By putting as much as possible in user space, we have achieved a flexible system without sacrificing performance.

In the Amoeba design, concessions to existing operating systems and software were carefully avoided. But a Unix emulation service was developed to run existing software on Amoeba.

Communication

Amoeba's conceptual model is that of a client thread (thread of control or lightweight process) performing operations on objects. For example, a common operation on a file object is reading data from it. Operations are implemented by making

Transport interface

The transport interface for the server consists of the calls `get_request` and `send_reply`, as described in the section on communication. They are generally part of a loop that accepts messages, does the work, and sends back replies, as in this C fragment:

```
/* Code for allocating a request buffer */
do {
    get_request(
        &port,
        &reqheader,
        &reqbuffer,
        reqbuflen);
    /* Code for unmarshalling
     * the request parameters
     */
    /* Call the implementation routine */
    /* Code for marshalling the

* reply parameters
*/
    send_reply(
        &repheader,
        &repbuffer,
        repbuflen);
    } while (1);

    Get_request blocks until a request comes in. Put_reply
    blocks until the header and buffer parameters can be
    reused. A client sends a request and waits for a reply by
    calling

    do_operation(reqheader, reqbuffer, reqbuflen,
        repheader, repbuffer, repbuflen);

    All of this code is generated automatically by the AIL com-
    piler from the object and operation descriptions given to it.
```

remote procedure calls.⁷ A client sends a request message to the service that manages the object. A server thread accepts the message, carries out the request, and sends the client a reply. To increase performance and fault tolerance, multiple server processes often jointly manage a collection of similar objects to provide a service.

Remote procedure calls. The kernel provides three basic system calls to user processes: `do_operation`, `get_request`, and `send_reply`. The first is used by clients to get work done. It consists of sending a message to a server and then blocking until a reply comes back. The second is used by servers to announce their willingness to accept messages addressed to a specific port. Servers use the third call to send replies back. All communication in Amoeba takes this form: First a client sends a request to a server; then the server accepts the request, does the work, and sends back the reply.

No doubt systems programmers would be content with only these three system calls, but for most applications programmers they are far too primitive. Therefore a more user-oriented interface has been built on top of the mechanism, to allow users to think directly in terms of objects and operations on these objects.

Corresponding to each type of object is a *class*. Classes can be composed hierarchically; that is, a class can contain operations from one or more underlying classes.

This multiple-inheritance mechanism allows many services to inherit the same interfaces for simple object manipulations, such as for changing the protection properties on an object or deleting it. The mechanism also allows all servers manipulating objects with filelike properties to inherit the same interface for low-level file I/O (read, write, append — see sidebar on Amoeba Interface Language). The mechanism resembles the filelike properties of Unix pipe and device I/O: The Unix read and write system calls can be used on files, terminals, pipes, tapes, and other I/O devices. But for more detailed manipulation, specialized calls are available (`ioctl`, `popen`, and so forth).

Remote procedure call transport. The Amoeba Interface Language compiler generates code to marshal or unmarshal the parameters of remote procedure calls into and out of message buffers and then call the Amoeba transport mechanism for delivery of request and reply messages (see sidebar on the transport interface). Messages consist of a header and a buffer. The header has a fixed format and contains addressing information (including the capability of the object that the remote procedure call refers to), an operation code that selects the function to be called on the object, and some space for additional parameters. The buffer can contain data. A file read or write call, for instance, uses the message header for the operation code plus

the length and offset parameters, and the buffer for the file data. With this setup, marshalling the file data (a character array) takes zero time because the data can be transmitted directly from and to the arguments specified by the program.

Locating objects. Before a request for an operation on an object can be delivered to a server thread that manages the object, such a thread must be located. All capabilities contain a service port field, which identifies the service that manages the object referred to by the capability. When a server thread makes a `get_request` call, it provides its service port to the kernel, which records it in an internal table. When a client thread calls `do_operation`, the kernel's job is to find a server thread with an outstanding `get_request` that matches the port in the capability provided by the client.

We call the process of finding the address of such a server thread *locating*. It works as follows: When a `do_operation` call comes into a kernel, a check is made to see if the port in question is already known. If not, the kernel broadcasts a special locate packet onto the network asking if anyone has an outstanding `get_request` for the port in question. If one or more kernels have servers with outstanding `get_requests`, they respond by sending their network addresses. The kernel doing the broadcasting records the port/network address pair in a cache for future use.

Secure communication

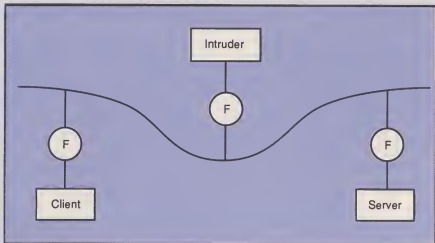
Client requests, addressed using an object's capability, are delivered to one of the servers with outstanding `get_request` calls on the capability's port. Ports consist of large, 48-bit numbers known only to the server processes that make up the service and to the server's clients. For a public service such as the file system, the port will be known to all users. The ports used by an ordinary user process will, in general, be kept secret. Knowledge of a port is taken by the system as *prima facie* evidence that the sender has a right to communicate with the service. Of course, the service is not required to carry out work for clients just because they know the port. For example, the file server will refuse to read or write files for clients lacking appropriate file capabilities. Thus Amoeba has two levels of protection: ports for protecting access to servers and capabilities for protecting access to individual objects.

Although the port mechanism conveniently handles partial authentication of clients ("if you know the port, you may at least talk to the service"), it does not authenticate servers. How do we ensure that malicious users do not make `get_request` calls on the file server's port and try to impersonate the file server to the rest of the system?

One approach is to have all ports manipulated by kernels that are presumed to be trustworthy and are supposed to know who may listen on which port. We have rejected this strategy because on some machines — for example, PCs — users might be able to tamper with the operating system kernel. Also, we believe that by making the kernel as small as possible, we can enhance system reliability as a whole. Therefore, we have chosen a different solution that can be implemented in either hardware or software.

In the hardware solution we place a small interface box, a function box or F-box, between each processor module and the network. The most logical place is on the VLSI chip used to interface to the network. Alternatively, it can be put on a small printed circuit board inside the wall socket through which PCs attach to the network. Where the processors have user mode and kernel mode, and the operating systems can be trusted, it can be put into the operating system. This is the solution in the current Amoeba implementation.

In the software solution we build the F-box with cryptographic algorithms, giving the same functional effect as the hardware box. In both cases we assume that all messages entering and leaving every processor undergo a simple transforma-



Clients, servers, intruders, and F-boxes.

tion that users cannot bypass.

The transformation works like this. Each port is really a pair of ports, P and G , related by $P = F(G)$, where F is a (publicly known) one-way function¹ performed by the F-box. The one-way function has the property that given G it is a straightforward computation to find P , but that given P , finding G is not feasible.

Using the one-way F-box, servers can be authenticated simply, as the figure illustrates. Each server chooses a get-port G and computes the corresponding put-port P . The get-port is kept secret; the put-port is distributed to potential clients or, in the case of public servers, is published. When the server is ready to accept client requests, it does a `get_request(G, ...)`. The F-box then computes $P = F(G)$ and waits for messages containing P to arrive. When one arrives, it is given to the server process. To send a message to the server, the client merely does `do_operation(P, ...)`, which sends a message containing P in a header field to the server. The F-box on the sender's side does not perform any transformation on the P field of the outgoing message.

Now consider the system from an intruder's point of view. To impersonate a server, the intruder must do `get_request(G, ...)`. However, G is a well-kept secret and is never transmitted on the network. Since we have assumed that G cannot be deduced from P (the one-way property of F) and that the F-box cannot be circumvented, intruders cannot intercept messages not intended for them. An intruder doing `get_request(P, ...)` will simply cause his F-box to listen to the (useless) port $F(P)$. Replies from the server to the

client are protected the same way, only with the client picking a get-port for the reply, say G' , and including $P' = F(G')$ in the request message.

The F-box makes it easy to implement digital signatures for further authentication, if that is desired. Each client chooses a random signature S and publishes $F(S)$. The F-box must be designed to work as follows. Each message presented to the F-box for transmission contains three special header fields: destination (P), reply (G'), and signature (S). The F-box applies the one-way function to the second and third of these, transmitting the three ports as P , $F(G')$, and $F(S)$, respectively. The first is used by the receiver's F-box to admit only those messages for which the corresponding get has been done, the second is used as the put-port for the reply, and the third can be used to authenticate the sender, since only the true owner of the signature will know what number to put in the third field to ensure that the publicly known $F(S)$ comes out.

The F-box implements security and protection simply, but gives operating system designers considerable latitude in choosing policies. The mechanism is flexible and general, so putting it into hardware should not preclude yet-to-be-designed operating systems.

Reference

1. M.V. Wilkes, *Time-Sharing Computer Systems*, 2nd ed., Elsevier, New York, 1969, pp. 129-132.

Table 1. The delay in milliseconds and the bandwidth in Kbytes per second for remote procedure calls between user processes in three common cases with three different systems. For local RPCs the client and server run on the same processor. The Unix driver implements Amoeba RPCs under Sun Unix.

	Delay (ms)			Bandwidth (Kbytes per second)		
	Case 1 (4 bytes)	Case 2 (8 Kbytes)	Case 3 (30 Kbytes)	Case 1 (4 bytes)	Case 2 (8 Kbytes)	Case 3 (30 Kbytes)
Native Amoeba local	0.8	2.5	7.1	5.0	3,277	4,255
Native Amoeba remote	1.4	13.1	44.0	2.9	625	677
Unix driver local	4.5	10.0	32.0	0.9	819	938
Unix driver remote	7.0	36.4	134.0	0.6	225	224
Sun RPC local	10.4	23.6	imposs.	0.4	347	imposs.
Sun RPC remote	12.2	40.6	imposs.	0.3	202	imposs.

Another broadcast is needed only if a server dies or migrates.

When Amoeba is run over a wide area network with a huge number of machines, a slightly different scheme is used. Each server wishing to export its service sends a special message to all domains where it wants its service known. (A domain could be a company, campus, city, or country.) In each domain a dummy process called a server agent is created. This process does a `get_request` using the server's port and then lies dormant until a request comes in. Then it forwards the message to the server for processing. Note that a port is just a randomly chosen 48-bit number. It does not identify a particular domain, network, or machine (see sidebar on secure communication).

Performance. We measured the speed of the Amoeba remote procedure call with some timing tests. For example, we booted the Amoeba kernel on two 16.7-megahertz Motorola MC68020s, created a user process on each, and let them communicate over a 10-megabit-per-second Ethernet. For a message consisting of just a header (no data), the complete remote procedure call (RPC) took 1.4 ms. With 8 Kbytes of data it took 13.1 ms, and with 30 Kbytes it took 44.0 ms. The latter corresponds to a throughput of 5.4 megabits per second, which is half the theoretical capacity of the Ethernet and much faster than the speeds most other systems achieve. Five client-server pairs together can achieve a total

throughput of 8.4 megabits per second, not counting Ethernet and Amoeba packet headers. Table 1 shows the speeds and throughput of local communication (communication between processes on the same machine) and remote communication (communication over Ethernet between processes on different machines). Remote operations were carried out with requests containing 4 bytes, 8 Kbytes, 30 Kbytes, and empty replies. Three RPC implementations were measured: RPCs on native Amoeba, the same Amoeba protocol used from a driver under Sun Unix, and Sun's own RPCs.

Why did we base the design on objects, capabilities, and RPCs? Objects are a natural way to program. By encapsulating information, users are forced to pay attention to precise interfaces, while irrelevant information is hidden from them. Capabilities are a clean and elegant way to name and protect objects. Using an encryption scheme to protect objects moves capability management out of the kernel. The RPC is an obvious way to implement the request-reply nature of performing operations on objects.

File system

Capabilities form Amoeba's low-level naming mechanism, but they are hard for people to use. Therefore an extra level of mapping is provided from symbolic hierarchical path names to capabilities. A typical user has access to literally thousands of

capabilities — those of the user's own private objects, but also capabilities of public objects, such as the executables of commands, pool processors, databases, and public files.

While a user could perhaps store his own private capabilities somewhere, a system manager or project coordinator cannot hand out capabilities explicitly to every user who may access a shared public object. Public places are needed where users can find capabilities of shared objects, so that when a new object is made shareable, or when a shareable object changes, its capability need be put in only one place.

Hierarchical directory structure. Hierarchical directory structures are ideal for implementing partially shared name spaces. Objects shared among members of a project team can be stored in a directory that only team members have access to. When directories are implemented as ordinary objects with a capability that is needed to use them, group members can be given access by giving them the capability of the directory, while others are denied access by withholding the capability. A directory capability is thus a capability for many other capabilities.

To a first approximation, a directory is a set of name/capability pairs. The basic operations on directory objects are lookup, enter, and delete. The first operation looks up an object name in a directory and returns its capability. The other two operations enter and delete objects from direct-

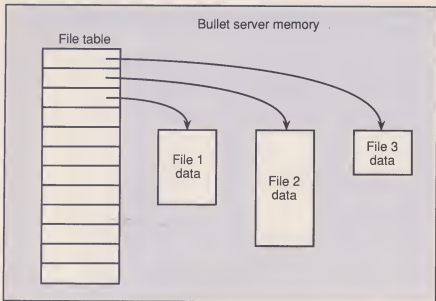


Figure 3. Bullet server file representation.

ries. Since directories themselves are objects, a directory can contain capabilities for other directories, thus allowing users to build an arbitrary graph structure.

Complex sharing can be achieved by making directories more sophisticated than we have just described. In reality, a directory is an $(n+1)$ -column table with ASCII names in column 0 and capabilities in columns 1 through n . A capability for a directory is really a capability for a specific column of a directory. Thus, for example, users could arrange their directories with one column for themselves, a second column for members of their group, and a third column for everyone else. This scheme provides the same protection rules as Unix, but obviously many other schemes are possible.

The directory service can be set up so that whenever a new object is entered in a directory, the directory service first asks the service managing the object to make n replicas, which can be physically distributed for reliability. All the capabilities are then entered into the directory.

Bullet service. The bullet service is a highly unusual file server. Each bullet server supports only three principal operations: read file, create file, and delete file.

When a file is created, the user normally provides all the data at once, creating the file and getting back a capability for it. In most circumstances the user will immediately give the file a name and ask the

directory service to enter the name/capability pair in some directory.

All files are immutable; once created, they cannot be changed. No write operation is supported. Since files cannot change, the directory service can replicate them at its leisure for redundancy.

Since the final file size is known when a file is created, files can be and are stored contiguously, both on the disk and in bullet servers' caches, as Figure 3 illustrates. Administrative information for a file is thus reduced to its origin and size, plus some ownership data. The complete administrative table is loaded into the bullet server's memory when it is booted. For a read operation the object number in the capability is used as an index into this table, and the file is read into the cache in a single (possibly multitrack) disk operation.

The bullet file service can deliver large files from its cache or accept large files into its cache at maximum RPC speeds, that is, at 677 Kbytes per second. A remote client can read a 4-Kbyte file from a bullet server's cache (over Ethernet) in 7 ms; a 1-Mbyte file takes 1.6 seconds.⁸

Although the bullet service wastes some space because of fragmentation, its performance easily compensates for having to buy an 800-Mbyte disk to store, say, 500 Mbytes of data.

Atomicity. Ideally, names always refer to consistent objects, and sets of names

always refer to mutually consistent sets of objects. In practice, this is seldom the case and is, in fact, not always necessary or desirable. But in many cases consistency is necessary.

Atomic actions are useful for achieving consistent updates to object sets. Protocols for atomic updates are well understood, and it is possible to provide a tool kit that allows independently implemented services to collaborate in atomic updates of multiple objects managed by several services.

For Amoeba we chose a different approach. The directory service handles atomic updates by allowing atomic changes in the mapping of arbitrary name sets onto arbitrary capability sets. The objects referred to by these capabilities must be immutable, either because the services that manage them refuse to change them (for example, the bullet service) or because the users refrain from changing them.

The atomic transactions provided by the directory service are not particularly useful for dedicated transaction-processing applications (for example, banking and airline reservation systems), but they do prevent the glitches that sometimes result when people use an application just as a new version is installed, or the lost update that results when two people simultaneously update a file.

Reliability and security. The directory service is crucial to the system: Nearly every application depends on it for finding the capabilities it needs. If the directory service stops, everything else will come to a halt as well. So that no single-site failure can bring it down, the directory service uses techniques similar to those used in fault-tolerant database systems to replicate all its internal tables on multiple disks.

The directory service must also work correctly and should never divulge a capability to an entity not entitled to see it. Yet even a perfectly designed directory service might allow unauthorized users to catch glimpses of data. Hardware diagnostic software, for example, has access to the directory server's disk storage. Bugs in the operating system kernel might allow users to read portions of the disk.

Directories can be encrypted so that bugs in the directory server and the operating system (or other idiosyncrasies) will not reveal confidential information. The encryption key can be exclusive-ORed with a random number and the result stored alongside the directory, while the random

number is put in the directory's capability. After giving the capability to the owner, the directory service itself can forget the random number. It needs the number only when the directory has to be decrypted to carry out operations on the directory, and will always receive the number in the capability that comes with every client's request.

Why did we design such an unconventional file system? Partly to achieve great speed and partly for simplicity in design and implementation. The use of immutable files (and some other objects) allows the replication mechanism to be centralized in the directory service. Immutable files are also easy to cache (because a cached immutable file can never become stale), an important issue when Amoeba is run over wide area networks.

Process management

Amoeba processes can have multiple threads of control. A process consists of a segmented virtual address space and one or more threads. Processes can be remotely created, destroyed, checkpointed, migrated, and debugged.

On a uniprocessor, threads run quasi-parallel; on a shared-memory multiprocessor, as many threads can run simultaneously as there are processors. Processes cannot be split over more than one machine.

Processes have explicit control over their address space. They can add new segments to it by mapping them in and remove segments by mapping them out. Along with virtual address and length, a capability can be specified in a map operation. This capability must belong to a filelike object, which is read by the kernel to initialize the new segment. This allows processes to do mapped-file I/O.

When a segment is mapped out, it remains in memory, although no longer as part of the address space of any process. The unmap operation returns a capability for the segment, which can then be read and written like a file. One process can thus map a segment out and pass the capability to another process; the other process can then map the segment in again. If the processes are on different machines, the contents of the segment are copied (by one kernel doing read operations and the other kernel servicing them). On the same machine, the kernel can use shortcuts for the same effect.

A process is created by sending a pro-

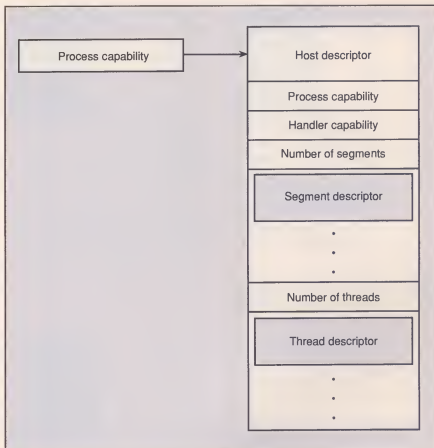


Figure 4. Layout of a process descriptor.

cess descriptor to a kernel in an execute-process request. Figure 4 shows the four parts of a process descriptor. The host descriptor describes the machine on which the process can run — for example, its instruction set, extended instruction sets (when required), and memory needs — but it can also specify a class of machines, a group of machines, or a particular machine. A kernel that does not match the host descriptor will refuse to execute the process.

The capabilities are next. One is the process capability that every client manipulating the process needs. Another is the capability of a handler, a service that deals with process exits, exceptions, signals, and other anomalies of the process.

The memory map has an entry for each segment in the address space of the process to be. An entry gives virtual address, segment length, how the segment should be mapped (read only, read/write, execute only, and so forth), and the capability of a file or segment from which the new segment should be initialized.

The thread map describes the initial state of each thread in the new process: the processor status word, the program counter, the stack pointer, the stack base, the register values, and the system call state. This rather elaborate notion of thread state allows process descriptors to be used not only for the representation of executable files, but also for processes being migrated, debugged, or checkpointed.

In most operating systems, system call state is large and complicated to represent outside an operating system kernel. In Amoeba, fortunately, there are very few system calls that can block in the kernel. The most complicated ones are for communication: `do_operation` and `get_request`.

Processes can be in two states: running or stunned. A stunned process — for example, a process being debugged — exists but does not execute instructions. The low-level communication protocols in the operating system kernel respond with “this process is stunned” messages to attempts to communicate with the process. The

sending kernel will keep trying to communicate until the process is running again or until it is killed. Thus, communication continues with a process being interactively debugged.

A running process can be stunned by a stun request from the outside world (the stunner must have the process capability as evidence of ownership) or by an uncaught exception. When the process becomes stunned, the kernel sends its state in a process descriptor to a handler, whose identity is a capability that belongs to the process' state. After examining the process descriptor, and possibly modifying it or the stunned process' memory, the handler can reply with either a resume or a kill command.

Debugging and migration are done through stunning. The debugger takes the role of the handler. For migration, first the candidate process is stunned; then the handler gives the process descriptor to the new host. The new host fetches memory contents from the old host in a series of file read requests, starts the process, and returns the capability of the new process to the handler. Finally, the handler returns a kill reply to the old host. Processes communicating with a process being migrated will receive "process is stunned" replies to their attempts until the process on the old host is killed. They will then get a "process not here" reaction. After they find the process on its new host, communication will resume.

The mechanism allows command interpreters to cache process descriptors of the programs they start and kernels to cache code segments of the processes they run. Combined, these caching techniques shorten process start-up times.

Our process management mechanisms are unusual, but they are intended for an unusual environment, one where remote execution is normal and local execution is the exception. The boundary conditions for our design were a few simple mechanisms that allowed us to implement process execution, migration, debugging, and checkpointing efficiently.

Unix emulation

Amoeba's system interface is quite different from those of today's popular operating systems. We did not want to write hundreds of utility programs for Amoeba from scratch, so we quickly decided to write a Unix emulation package to allow

most Unix utilities to work on Amoeba, sometimes with small changes. We considered binary compatibility but rejected it for an initial emulation package because binary compatibility is more complicated and less useful. (First, we would have to choose a particular version of Unix; second, binaries usually work for only one machine architecture, while sources can be compiled for any machine architecture; and third, binary emulation is bound to be slow.)

Our emulation facility started as a library of Unix routines that have the standard Unix interface and semantics but do their work by calling the bullet service, the directory service, and the Amoeba process management facilities. The system calls implemented initially were those for file I/O (open, close, dup, read, write, lseek) and a few of the ioctl calls for ttys. These were very easy to implement under Amoeba (about two weeks' work) and were enough to run a surprising number of Unix utilities.

Next a session server was developed to allocate Unix PIDs and PPIDs, and to assist in the handling of system calls involving them (for example, fork, exec, signal, kill). The session server is also used for dealing with Unix pipes and allows many other Unix utilities to run on Amoeba. Users each start one session server alongside their login shell.

About 150 utilities now run on Amoeba without any changes to the source code. We have not attempted to port some of the more esoteric Unix programs, but we are working to make our Unix interface compatible with some emerging standards (for example, IEEE Posix).

The X Window System has been ported to Amoeba and supports both TCP/IP and Amoeba RPCs, so an X client on Amoeba can converse with an X server on Amoeba and vice versa.

The Unix utilities have eased the transition to Amoeba. Gradually, however, many of them will be replaced by utilities better adapted to the Amoeba distributed environment. Our new parallel Make is an obvious example.

If we had designed a system that was binary compatible with Unix, it would not have been much of a step beyond the ideas of the early 1970s. We wanted a new system for the 1990s, designed from the ground up. If the Unix designers had constrained themselves to being binary compatible with the then-popular RT-11 operating system, Unix would not be where it is now.

Among the design decisions for Amoeba we have been most pleased with is our determination not to restrict ourselves to existing operating systems or operating system interfaces. Unix is an excellent operating system, but it was not designed for distributed systems. We could not have made such a balanced design with a Unix interface. Nevertheless, we found it remarkably easy to port to Amoeba all the Unix software we wanted to use. Programs that are hard to port are mostly for operations that Amoeba handles in other ways (network access and system maintenance and management, for example).

Amoeba's use of objects and capabilities means that when we design a service we need not worry about the protection of its objects. The capabilities mechanism automatically provides enough protection. The system also provides a very uniform and decentralized object-naming and object-access mechanism.

Building directly on the hardware instead of on an existing operating system has been absolutely essential to Amoeba's success. A primary goal was to design and build a high-performance system, and this can hardly be done on top of another system. As far as we can tell, only systems with custom-built hardware or special microcode can outperform Amoeba's remote procedure calls and file system on comparable hardware.

The Amoeba kernel is small and simple. It implements only a few operations for process management and interprocess communication, but they are versatile and easy to use. The kernel is easy to port between hardware platforms. Amoeba now runs on VAXs and on Motorola MC68020 and MC68030 processors, and is currently being ported to the Intel 80386. ■

Acknowledgments

The work described here has been supported by grants from NWO, the Netherlands Organization for Scientific Research; SION, the Foundation for Computer Science Research in the Netherlands; OSF, the Open Software Foundation; and Digital Equipment Corporation.

References

1. R.M. Needham and A.J. Herbert, *The Cambridge Distributed Computing System*, Addison-Wesley, Reading, Mass., 1982.
2. B. Walker et al., "The LOCUS Distributed Operating System," *Proc. Ninth Symp. Operating System Principles*, ACM, *Operating Systems Review*, Vol. 17, No. 5, 1983, pp. 49-70.
3. M. Accetta et al., "Mach: New Kernel Foundation for UNIX Development," *Proc. Summer Usenix Conference*, Usenix, Sunset Beach, Calif., 1986.
4. D.R. Cheriton, "The V Distributed System," *Comm. ACM*, Vol. 31, No. 3, Mar. 1988, pp. 314-333.
5. M. Rozier et al., "CHORUS Distributed Operating Systems," Report CS/Tech. Report-88-7.6, Chorus Systems, Paris, 1988.
6. S.J. Mullender and A.S. Tanenbaum, "The Design of a Capability-Based Distributed Operating System," *Computer J.*, Vol. 29, No. 4, Mar. 1986, pp. 289-300.
7. A.D. Birrell and B.J. Nelson, "Implementing Remote Procedure Calls," *ACM Trans. Computer Systems*, Vol. 2, No. 1, Feb. 1984, pp. 39-59.
8. R. van Renesse, J.M. van Staveren, and A.S. Tanenbaum, "Performance of the Amoeba Distributed Operating System," *Software — Practice and Experience*, Vol. 19, No. 3, Mar. 1989, pp. 223-234.



Guido van Rossum is a research assistant at the Centre for Mathematics and Computer Science in Amsterdam. Since 1987 he has been with the Amoeba project, working on an RPC interface specification language, a Unix emulation facility, user interface issues, and system integration. Earlier he worked on the ABC Programming Language project.

Van Rossum studied mathematics and computer science at the University of Amsterdam and received a master's degree in 1982.



Hans van Staveren is one of the implementers of the Amoeba distributed operating system, working primarily on network protocols and kernel efficiency. Earlier he spent four years researching code generation in the framework of the Amsterdam Compiler Kit.

Van Staveren graduated from the Free University in Amsterdam in 1980.



Andrew S. Tanenbaum is a professor of computer science at the Free University in Amsterdam. His research interests include distributed operating systems, programming languages, and compilers. He is the author of the Minix operating system, a principal designer of the Amsterdam Compiler Kit, and a chief architect of the Amoeba distributed operating system.

Tanenbaum received his BS from MIT and his PhD from the University of California, Berkeley. He is a member of ACM, the IEEE Computer Society, and Sigma Xi.



Sape J. Mullender heads the distributed systems and computer networks research group at the Centre for Mathematics and Computer Science in Amsterdam. He has been a visiting scientist at DEC's Systems Research Center in California and a visiting research fellow at Cambridge University. Mullender's research interests include high-performance communication in distributed systems and the design of scalable fault-tolerant distributed file servers. He is also concerned with organization and protection in distributed systems that can span a continent.

Mullender is vice chairman and conference coordinator of the ACM Special Interest Group on Operating Systems. He received his PhD at the Free University in Amsterdam. He is a member of ACM and the IEEE.



Robbert van Renesse is a researcher in the Computer Science Department at Cornell University on a grant from the Netherlands Organization for Scientific Research. He is working on the management of distributed systems to improve their robustness, performance, and scalability.

Van Renesse received his PhD in computer science in 1989 from the Free University of Amsterdam.

The authors can be contacted at the Centre for Mathematics and Computer Science, PO Box 4079, 1009 AB Amsterdam, the Netherlands.

日英両国語に通じた「バイリンガル」な
テクニカル・コンサルタント募集

Bilingual Japanese-English Technical Consultants

Learning Tree International, world leader in advanced technology education, is now offering four-day intensive courses in Japan on Software Development, CASE, UNIX/C, Datacomm/Networks, Signal/Image Processing, Project Management and other computer-related topics. We need technical experts who are able to teach these short courses on a consulting basis in Tokyo in Japanese.

Please phone Dr. David Collins
at (213) 417-9700.



Learning Tree
International

6053 W. Century Blvd. / Los Angeles, CA 90045

Reader Service Number 4

Algorithms Implementing Distributed Shared Memory

Michael Stumm and Songnian Zhou
University of Toronto

Traditionally, communication among processes in a distributed system is based on the *data-passing model*. Message-passing systems or systems that support remote procedure calls (RPCs) adhere to this model. The data-passing model logically and conveniently extends the underlying communication mechanism of the system; port or mailbox abstractions along with primitives such as Send and Receive are used for interprocess communication. This functionality can also be hidden in language-level constructs, as with RPC mechanisms. In either case, distributed processes pass shared information by value.

In contrast to the data-passing model, the *shared memory model* provides processes in a system with a shared address space. Application programs can use this space in the same way they use normal local memory. That is, data in the shared space is accessed through Read and Write operations. As a result, applications can pass shared information by reference. The shared memory model is natural for distributed computations running on shared memory multiprocessors. For loosely coupled distributed systems, no physically shared memory is available to support such a model. However, a layer of software can provide a shared memory abstraction to the

This article compares several algorithms for implementing distributed shared memory. It shows that the performance of these algorithms is sensitive to the memory access behavior of applications.

applications. This software layer, which can be implemented either in an operating system kernel or, with proper system kernel support, in runtime library routines, uses the services of an underlying (mes-

sage passing) communication system. The shared memory model applied to loosely coupled systems is referred to as *distributed shared memory*.

In this article, we describe and compare basic algorithms for implementing distributed shared memory by analyzing their performance. Conceptually, these algorithms extend local virtual address spaces to span multiple hosts connected by a local area network, and some of them can easily be integrated with the hosts' virtual memory systems. In the remainder of this section, we describe the merits of distributed shared memory and the assumptions we make with respect to the environment in which the shared memory algorithms are executed. We then describe four basic algorithms, provide a comparative analysis of their performance in relation to application-level access behavior, and show that the correct choice of algorithm is determined largely by the memory access behavior of the applications. We describe two particularly interesting extensions of the basic algorithms and conclude by observing some limitations of distributed shared memory.

Advantages of distributed shared memory. The primary advantage of distributed shared memory over data passing

is the simpler abstraction provided to the application programmer, an abstraction the programmer already understands well. The access protocol used is consistent with the way sequential applications access data, allowing for a more natural transition from sequential to distributed applications. In principle, parallel and distributed computations written for a shared memory multiprocessor can be executed on a distributed shared memory system without change. The shared memory system hides the remote communication mechanism from the processes and allows complex structures to be passed by reference, substantially simplifying the programming of distributed applications. Moreover, data in distributed shared memory can persist beyond the lifetime of a process accessing the shared memory.

In contrast, the message-passing models force programmers to be conscious of data movement between processes at all times, since processes must explicitly use communication primitives and channels or ports. Also, since data in the data-passing model is passed between multiple address spaces, it is difficult to pass complex data structures. Data structures passed between processes must be marshaled and unmarshaled by the application. (Marshaling refers to the linearizing and packing of a data structure into a message.)

For these reasons, the code of distributed applications written for distributed shared memory is usually significantly shorter and easier to understand than equivalent programs that use data passing.

The advantages of distributed shared memory have made it the focus of recent study and have prompted the development of various algorithms for implementing the shared data model.¹⁻⁸ Several implementations have demonstrated that, in terms of performance, distributed shared memory can compete with direct use of data passing in loosely coupled distributed systems.^{2,4,9}

In a few cases, applications using distributed shared memory can even outperform their message-passing counterparts (even though the shared memory system is implemented on top of a message-passing system). This is possible for three reasons:

(1) For shared memory algorithms that move data between hosts in large blocks, communication overhead is amortized over multiple memory accesses, reducing overall communication requirements if the application exhibits a sufficient degree of locality in its data accesses.

(2) Many (distributed) parallel applications execute in phases, where each computation phase is preceded by a data-exchange phase. The time needed for the data-exchange phase is often dictated by the throughput limitations of the communication system. Distributed shared memory algorithms typically move data on demand as they are being accessed, eliminating the data-exchange phase, spreading the communication load over a longer period of time, and allowing for a greater degree of concurrency.

(3) The total amount of memory may be increased proportionally, reducing paging and swapping activity.⁴

Similar systems. Distributed shared memory systems have goals similar to those of CPU cache memories in shared-memory multiprocessors, local memories in shared memory multiprocessors with nonuniform memory access (NUMA) times, distributed caching in network file systems, and distributed databases. In particular, they all attempt to minimize the access time to potentially shared data that is to be kept consistent. Consequently, many of the algorithmic issues that must be addressed in these systems are similar.

Although these systems therefore often use algorithms that appear similar from a distance, their details and implementations can vary significantly because of differences in the cost parameters and in the ways they are used. For example, in NUMA multiprocessors, the memories are physically shared and the time differential between accesses of local and remote memory is lower than in distributed systems, as is the cost of transferring a block of data between the local memories of two processors. Hence, some of the algorithms we discuss in this article will be relevant to designers of NUMA memory management systems, but the algorithms they chose as most appropriate may differ.

Similarly, in bus-based shared memory multiprocessors, replication in the CPU caches (to avoid the much higher delays of accessing main memory and to reduce bus congestion) can be implemented cost effectively because of the reliability and broadcast properties of the bus. On the other hand, in distributed systems where communication is unreliable, we show that algorithms without replication can benefit certain types of applications. As a third example, distributed file systems and databases must provide for persistent data and, in the case of distributed databases, reliability and atomicity. These requirements

can significantly affect the choice of algorithm.

Model and environmental assumptions. In our discussions and analyses, we make certain assumptions with respect to the environment in which the algorithms are implemented. These are described here. The extrapolation of our results to other environments is left for future work.

In general, the performance of distributed and parallel applications is dictated primarily by communication costs, which in turn are dictated by the underlying hardware. Here, we assume a distributed system environment consisting of a cluster of hosts connected by a local area network, such as an Ethernet. In this environment, communication between processors is unreliable and slow relative to local memory access. We assume that broadcast and multicast communication, where a single message can be sent (unreliably) to multiple receivers in a single network transaction, is available. Most bus and ring networks fit this description.

For performance analysis, communication costs are abstracted in terms of the number of messages sent and the number of packet events. A packet event is the cost associated with either receiving or sending a small packet (about 1 millisecond on a Sun-3/50). A point-to-point message transfer therefore requires one message, a packet event at the sending site, and a packet event at the receiving site. A multicast or broadcast message transmission requires one message, a packet event at the sending site, and a packet event at each receiving site.

The shared memory model provides two basic operations for accessing shared data:

```
data := read( address )  
write( data, address )
```

Read returns the data item referenced by Address, and Write sets the contents referenced by Address to the value of Data. For simplicity, the algorithms for implementing shared data are described in terms of these two operations. We assume that the distributed applications call these functions explicitly, although this may not always be necessary with suitable compiler and/or operating-system support, and that the data item accessed is always a single word.

Of course, variations in the syntax and semantics of these operations are possible. For instance, the operations may be called by a number of different names, such as

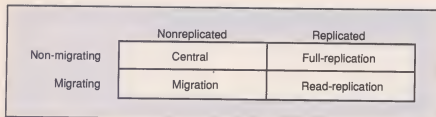


Figure 1. Four distributed memory algorithms.

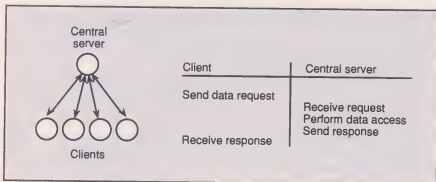


Figure 2. The central-server algorithm.

fetch/store and in/out. The type of the data being accessed can also vary and include integers, byte arrays, or more complex user-defined types. Finally, the semantics of the memory access functions can go beyond those offered by traditional memory systems and can include atomic enqueueing or dequeuing operations, or even entire database operations. For example, Linda¹⁰ is a programming language that directly supports a shared data space by providing a shared tuple space and three basic operations: Read reads an existing data item called "tuple" from the tuple space, Out adds a new tuple, and In reads and then removes an existing tuple. Linda's tuples are addressed by content rather than by location.

Basic algorithms

This section describes four basic distributed shared memory algorithms. For each of these algorithms, we consider the cost of read and write operations and issues in their implementation. The algorithms described can be categorized by whether they migrate and/or replicate data, as depicted in Figure 1. Two of the algorithms migrate data to the site where it is accessed in an

attempt to exploit locality in data accesses and decrease the number of remote accesses, thus avoiding communication overhead. In other words, processes should not be able to observe (by reading and writing shared data) that all data accesses are not directed to the same copy of data.

Implementations of distributed shared memory based on replication should make this replication transparent to the applications. In other words, processes should not be able to observe (by reading and writing shared data) that all data accesses are not directed to the same copy of data.

More formally,¹¹ the result of applications using shared data should be the same as if the memory operations of all hosts were executing in some sequential order, and the operations of each individual host appear in sequence in the order specified by its program, in which case the shared memory is said to be *consistent*.

Shared memory in a shared memory multiprocessor is expected to behave this way. This definition of consistency should not be confused with a stricter definition requiring read accesses to return the value of the most recent write to the same location, which is naturally applicable to concurrent processes running on a uniprocessor but not necessarily to those on shared

memory multiprocessors with CPU caches and write-back buffers.¹² If the stricter definition holds, then so does the weaker definition (but the converse is not true).

Central-server algorithm. The simplest strategy for implementing distributed shared memory uses a central server that is responsible for servicing all accesses to shared data and maintains the only copy of the shared data. Both read and write operations involve the sending of a request message to the data server by the process executing the operation, as depicted in Figure 2. The data server executes the request and responds either with the data item in the case of a read operation or with an acknowledgment in the case of a write operation.

A simple request-response protocol can be used for communication in implementations of this algorithm. For reliability, a request is retransmitted after each time-out period with no response. This is sufficient, since the read request is idempotent; for write requests, the server must keep a sequence number for each client so that it can detect duplicate transmissions and acknowledge them appropriately. A failure condition is raised after several timeout periods with no response.

Hence, this algorithm requires two messages for each data access: one from the process requesting the access to the data server and the other containing the data server's response. Moreover, each data access requires four packet events: two at the requesting process (one to send the request and one to receive the response), and two at the server.

One potential problem with the central server is that it may become a bottleneck, since it has to service the requests from all clients. To distribute the server load, the shared data can be distributed onto several servers. In that case, clients must be able to locate the correct server for data access. A client can multicast its access requests to all servers, but this would not significantly decrease the load on all servers, since every server would incur the overhead of a packet event for each such request. A better solution is to partition the data by address and use some simple mapping function to decide which server to contact.

Migration algorithm. In the migration algorithm, depicted in Figure 3, the data is always migrated to the site where it is accessed. This is a "single reader/single writer" (SRSW) protocol, since only the threads executing on one host can read or

write a given data item at any one time.

Instead of migrating individual data items, data is typically migrated between servers in a fixed size unit called a *block* to facilitate the management of the data. The advantage of this algorithm is that no communication costs are incurred when a process accesses data currently held locally.

If an application exhibits high locality of reference, the cost of data migration is amortized over multiple accesses. However, with this algorithm, it is also possible for pages to *thrash* between hosts, resulting in few memory accesses between migrations and thereby poor performance. Often, the application writer will be able to control thrashing by judiciously assigning data to blocks.

A second advantage of the migration algorithm is that it can be integrated with the virtual memory system of the host operating system if the size of the block is chosen equal to the size of a virtual memory page (or a multiple thereof). If a shared memory page is held locally, it can be mapped into the application's virtual address space and accessed using the normal machine instructions for accessing memory. An access to a data item located in data blocks not held locally triggers a page fault so that the fault handler can communicate with the remote hosts to obtain the data block before mapping it into the application's address space. When a data block is migrated away, it is removed from any local address space it has been mapped into.

The location of a remote data block can be found by multicasting a migration request message to all remote hosts, but more efficient methods are known.³ For example, one can statically assign each data block to a managing server that always knows the location of the data block. To distribute the load, the management of all data blocks is partitioned across all hosts. A client queries the managing server of a data block, both to determine the current location of the data block and to inform the manager that it will migrate the data block.

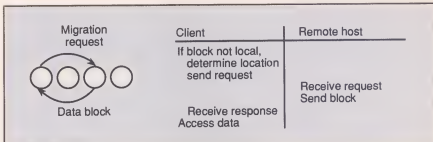


Figure 3. The migration algorithm.

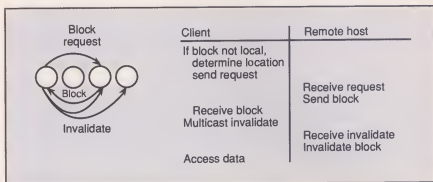


Figure 4. The write operation in the read-replication algorithm.

become more expensive, since the replicas may have to be invalidated or updated to maintain consistency. Nevertheless, if the ratio of reads over writes is large, the extra expense for the write operations may be more than offset by the lower average cost of the read operations.

Replication can be naturally added to the migration algorithm by allowing either one site a read/write copy of a particular block or multiple sites read-only copies of that block. This type of replication is referred to as "multiple readers/single writer" (MRSW) replication.

For a read operation on a data item in a block that is currently not local, it is necessary to communicate with remote sites to first acquire a read-only copy of that block and to change to read-only the access rights to any writable copy if necessary before the read operation can complete. For a write operation to data in a block that is either not local or for which the local host has no write permission, all copies of the same block held at other sites must be invalidated before the write can proceed. (See Figure 4.)

This strategy resembles the write-invalidate algorithm for cache consistency implemented by hardware in some multiprocessors.¹³ The read-replication algorithm is consistent because a read access always returns the value of the most recent write to the same location.

This type of replication has been investigated extensively.^{3,4} In Li's implementation, each block has a server designated as its *owner* that is responsible for maintaining a list of the servers having a read-only copy of that block. This list is called the block's *copy set*.

A read (or write) access to a block for which a server does not have the appropriate access rights causes a read (or write) fault. The fault handler transmits a request to the server that has ownership of the appropriate block. For a read fault, the owning server replies with a copy of the block, adds the requesting server to the copy set of the requested block and, if necessary, changes the access rights of its local copy to read-only.

When a write fault occurs, ownership for the block is transferred from the previ-

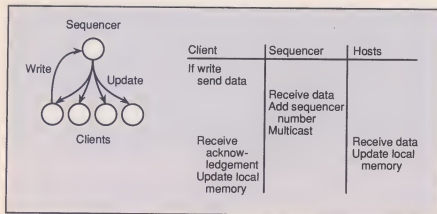


Figure 5. The full-replication algorithm.

ous owner to the server where the write fault occurred. After receiving the response, the write-fault handler requests all servers in the copy set to invalidate their local copy, after which the access rights to that block are set to write access at the site of the new owner and the copy set is cleared.

Full-replication algorithm. Full replication allows data blocks to be replicated even while being written to. The full-replication algorithm therefore adheres to a "multiple readers/multiple writers" (MRMW) protocol. Keeping the data copies consistent is straightforward for non-replicated algorithms, since accesses to data are sequenced according to the order in which they occur at the site where the data is held. In the case of fully replicated data, accesses to the data must either be properly sequenced or controlled to ensure consistency.

One possible way to keep the replicated data consistent is to globally sequence the write operations, while only sequencing the read operations relative to the writes that occur local to the site where the reads are executed. For example, the *write update* algorithm for cache consistency implemented by hardware in some multiprocessors¹³ maintains consistency in this fashion; that is, its reads occur locally from the cache while writes are broadcast over the bus that sequences them automatically.

A simple strategy based on sequencing uses a single global gap-free sequencer, depicted in Figure 5, which is a process executing on a host participating in distributed shared memory. When a process attempts a write to shared memory, the in-

tended modification is sent to the sequencer. This sequencer assigns the next sequence number to the modification and multicasts the modification with this sequence number to all sites.

Each site processes broadcast write operations in sequence number order. When a modification arrives at a site, the sequence number is verified as the next expected one. If a gap in the sequence numbers is detected, either a modification was missed or a modification was received out of order, in which case a retransmission of the modification message is requested. (This implies that somewhere a log of recent write requests be maintained.) In effect, this strategy implements a negative acknowledgment protocol.

In the common case within our assumed environment, packets arrive at all sites in their proper order. Therefore, a write requires two packet events at the writing process, two packet events at the sequencer, and a packet event at each of the other replica sites, for a system total of $5+2$ packet events with S participating sites.

Many variants to the above algorithms exist. For example, Bisiani and Forin¹ described an algorithm for full replication that uses the same principle as the sequencing algorithm to ensure individual data structures remain consistent. However, rather than using a single server to sequence all writes, writes to any particular data structure are sequenced by the server that manages the master copy of that data structure. Although each data structure is maintained in a consistent manner, there is no assurance with this algorithm that updates to multiple data structures are made consistently.

Performance comparisons

All four algorithms described in the previous section ensure consistency in distributed shared memory. However, their performance is sensitive to the data-access behavior of the application. In this section, we identify the factors in data-access costs and investigate the application behaviors that have significant bearings on the performance of the algorithms. Based on some simple analyses, we compare the relative merits of the algorithms in an attempt to unveil the underlying relationship between access patterns of applications and the shared memory algorithms that are likely to produce better performance for them.

Model and assumptions. The parameters in Figure 6 characterize the basic costs of accessing shared data and the application behaviors. Among them, the two types of access fault rates, f and f' , have the greatest impact on performance of the corresponding algorithms but, unfortunately, are also the most difficult to assess since they vary widely from application to application. We should also point out that these parameters are not entirely independent of one another. For instance, the size of a data block and therefore the block transfer cost, P , influences f and f' , in conflicting directions. As the block size increases, more accesses to the block are possible before another block is accessed; however, access interferences between sites become more likely. S also has direct impact on the fault rates. Nevertheless, the analyses below suffice to characterize the shared memory algorithms.

To focus on the essential performance characteristics of the algorithms and to simplify our analyses, we make a number of assumptions:

- (1) The amount of message traffic will not cause network congestion. Hence, we will only consider the message-processing costs, p and P , but not the network bandwidth occupied by messages.
- (2) Server congestion is not serious enough to significantly delay remote access. This is reasonable for the algorithms we study, since there are effective ways to reduce the load on the servers (see the "Central-server algorithm" section).
- (3) The cost of accessing a locally available data item is negligible compared to remote access cost and therefore does not

show up in our access cost calculations below.

(4) Message passing is assumed to be reliable, so the cost of retransmission is not incurred. Note, however, that the cost for acknowledgment messages, required to determine whether a retransmission is necessary or not, is included in our models.

To compare the performance of the distributed shared memory algorithms, we need to define a performance measure. Distributed shared memory is often used to support parallel applications in which multiple threads of execution may be in progress on a number of sites. We therefore choose the average cost per data access to the entire system as the performance measure. Hence, if a data access involves one or more remote sites, the message-processing costs on both the local and remote site(s) are included.

Using the basic parameters and the simplifying assumptions described above, the average access costs of the four algorithms can be expressed as follows:

$$\text{Central server: } C_c = (1 - \frac{1}{S}) * 4p$$

$$\text{Migration: } C_m = f * (2P + 4p)$$

$$\text{Read replication: } C_r = f' * (2P + 4p + \frac{Sp}{r+1})$$

$$\text{Full replication: } C_p = \frac{1}{r+1} * (S + 2)p$$

Each of these expressions has two components. The first component, to the left of the $*$, is the probability of an access to a data item being remote. The second component, to the right of the $*$, is equal to the average cost of accessing a remote data item. Since the cost of local accesses is assumed to be negligible, the average cost of accessing a data item therefore equals the product of these two components.

In the case of the central-server algorithm, the probability of accessing a remote data item is $1-1/S$, in which case four packet events are necessary for the access (assuming that data is uniformly distributed over all sites). The overall cost, C_c , is therefore mainly determined by the cost of a packet event, as long as the number of sites is more than four or five.

For the migration algorithm, f represents the probability of accessing a nonlocal data item. The cost of accessing a data item in that case equals the cost of bringing the data block containing the data item to the local site, which includes a total of one block transfer ($2P$) and four packet events distributed across the local, manager, and server sites. We assume here that the local, manager, and server sites are all distinct,

p : The cost of a packet event, that is, the processing cost of sending or receiving a short packet, which includes possible context switching, data copying, and interrupt handling overhead. Typical values for real systems range from one to several milliseconds.

P : The cost of sending or receiving a data block. This is similar to p , except that P is typically significantly higher. For an 8-kilobyte block, except of ten multiple packets are needed, typical values range from 20 to 40 milliseconds.

For our analyses, only the ratio between P and p is important, rather than their absolute values.

S : The number of sites participating in distributed shared memory.

r : Read/write ratio, that is, there is one write operation for every r reads on average. This parameter is also used to refer to the access pattern of entire blocks. Although the two ratios may differ, we assume they are equal in order to simplify our analyses.

f : Probability of an access fault on a nonreplicated data block (used in the migration algorithm). This is equal to the inverse of the average number of consecutive accesses to a block by a single site, before another site makes an access to the same block, causing a fault. f characterizes the locality of data accesses for the migration algorithm.

f' : Probability of an access fault on replicated data blocks used in the read-replication algorithm. It is the inverse of the average number of consecutive accesses to data items in blocks kept locally, before a data item in a block not kept locally is accessed. f' characterizes the locality of data accesses for the read-replication algorithm.

Figure 6. Parameters that characterize the basic costs of accessing shared data.

and that the request is forwarded by the manager to the server. The sequence of packet events is send (on local site), receive (on manager site), forward (on manager site), and receive (on server site).

For read replication, the remote access cost approximates that of the migration algorithm except that, in the case of a write fault (which occurs with a probability of $1/(r+1)$), a multicast invalidation packet must be handled by all S sites. The block transfer cost is always included in our expression, although it may not be necessary if a write fault occurs and a local (read) copy of the block is available.

Finally, for the full-replication algorithm, the probability of a remote access equals the probability of a write access. The associated cost for this write is always a message from the local site to the sequencer (two packet events), followed by a multicast update message to all other sites (S packet events).

Comparative analyses. The discussion above prepares us to make some pair-wise comparisons of the algorithms' performance to illustrate the conditions under which one algorithm might outperform another. Each comparison is made by equating the average costs of the two algorithms concerned, to derive a curve along which they yield similar performance.

This curve, which we call the *equal-performing curve*, divides the parameter space into two halves such that in each half one of the algorithms will perform better than the other. For example, in the following comparison between migration and read replication, the equation on the right of Figure 7 is that of the equal-performing curve, derived from $C_m = C_r$ (with some rearrangement). Since all of the cost formulas include packet cost p , only the ratio between P and p matters in the following comparative analyses. We assume the value of P/p to be 20. Based on these

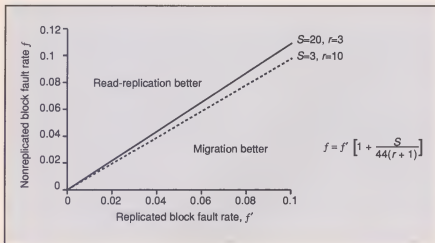


Figure 7. Performance comparison: migration versus read replication.

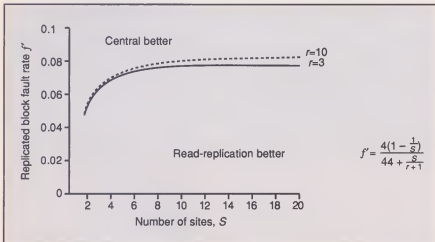


Figure 8. Performance comparison: central server versus read replication.

comparisons, we will be able to make some general comments on performance.

Migration versus read replication. The only difference between these two algorithms is that replication is used in the read-replication algorithm to allow interleaved reads on several sites without block movements, but at the expense of multicasting invalidation requests upon updates. Interestingly, as shown in Figure 7, the invalidation traffic does not have a strong influence on the algorithms' relative performances. As long as the cost of a block transfer is substantially higher than

that of a small message, the curves for different values of S and r cluster closely together and are very close to the $f=f'$ line.

Typically, read replication effectively reduces the block fault rate because, in contrast to the migration algorithm, interleaved read accesses to the same block will no longer cause faults, so the value of f' is smaller than f . Therefore, one can expect read replication to outperform migration for a vast majority of applications.

Central server versus read replication. Figure 8 compares the central-server and read-replication algorithms. The equal-

performing curve is almost flat, that is, insensitive to the number of sites. Moreover, the influence of the read/write ratio is also minimal. Hence, the key consideration in choosing between the two algorithms is the locality of access. Typically, a block fault rate of 0.07 (14 accesses between faults) is considered very high (faults very frequent). Therefore, read replication appears to be more favorable for many applications.

Read replication versus full replication. Both algorithms use read replication. The full-replication algorithm is more aggressive in that multiple copies are maintained even for updates. Figure 9 shows that the relative performance of the two algorithms depends on a number of factors, including the degree of replication, the read/write ratio, and the degree of locality achievable in read replication. The full-replication algorithm is not susceptible to poor locality, since all data is replicated at all sites. On the other hand, the cost of multicast increases with S . Therefore, full replication performs poorly for large systems and when update frequency is high (that is, when r is low).

Central server versus full replication. These two algorithms represent the two extremes in supporting shared data: one is completely centralized, the other is completely distributed and replicated. Except for small values of S , the curve shown in Figure 10 is almost linear. For S values of up to about 20, the aggressive replication of full replication seems to be advantageous, as long as the read/write ratio is five or higher. For very large replication, however, the update costs of full replication catch up, and the preference turns to the simple central-server algorithm.

Remaining comparisons. The two remaining pairs not yet considered are summarized as follows. The comparison between central server and migration resembles that between central server and read replication, with a rather flat curve beneath $f = 0.09$. Thus, unless the block fault rate is very high, migration performs better. The comparison between migration and full replication reveals no clear winner, as in the case of read replication versus full replication, with the key deciding factors being S and r .

Comments. Based on the comparisons above, we can make a few observations. The central-server algorithm is simple to implement and may be sufficient for infre-

quent accesses to shared data, especially if the read/write ratio is low (that is, a high percentage of accesses are writes). This is often the case with locks, as will be discussed further below. However, locality of reference and a high-block-hit ratio are present in a wide range of applications, making block migration and replication advantageous.

The fault rate of the simple migration algorithm may increase due to interleaved accesses if different data items that happen to be in the same block are accessed by different sites. It thus does not explore locality to its full extent. The full-replication algorithm is suitable for small-scale replication and infrequent updates.

In contrast, the read-replication algorithm is often a good compromise for many applications. The central-server and full-replication algorithms share the property of insensitivity to access locality, so they may outperform the read-replication algorithm if the application exhibits poor access locality.

A potentially serious performance problem with algorithms that move large data blocks is *block thrashing*. For migration, it takes the form of moving data back and forth in quick succession when interleaved data accesses are made by two or more sites. For read replication, it takes the form of blocks with read-only permissions being repeatedly invalidated soon after they are replicated.

Such situations indicate poor (site) locality in references. For many applications, shared data can be allocated and the computation can be partitioned to minimize thrashing. Application-controlled locks can also be used to suppress thrashing (see the section entitled "Application-level control with locking"). In either case, the complete transparency of the distributed shared memory is compromised somewhat.

Applications. From the above comparisons, it is clear that strong interactions exist between the shared data access patterns of applications and their expected performance using the various algorithms. To make our discussion more concrete, we study such interactions further in this section, based on our experience implementing the algorithms and measuring their performance. We do this by examining a few types of applications and the consistency algorithms that might suit them.

Numerical subroutines and other applications. The Blas-III package contains a

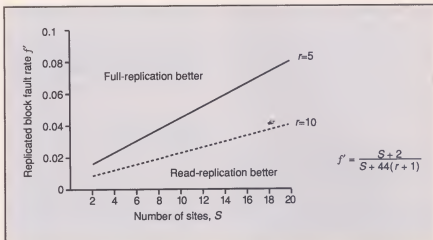


Figure 9. Performance comparison: read replication versus full replication.

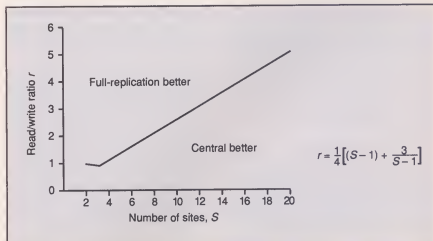


Figure 10. Performance comparison: central server versus full replication.

frequently used set of matrix manipulation subroutines implemented in Fortran. Typically, one or more (large) matrices are used as input to generate a result matrix of the same dimension. The amount of computation, usually substantial, can be sped up by assigning processors to compute subregions of the result matrix. This results in the input data being widely read-shared and the individual regions of the result matrix being updated by a single site.

Thus, read replication is highly desirable, whereas update multicast for write replication is unnecessary and too costly.

We observed excellent speedup using the read replication algorithm. Li studied similar applications in his thesis, and also reported very good speedup.³

Interestingly, this data-access pattern is widespread. For example, we converted a graphics application to use distributed shared memory. This application uses a three-dimensional description of a set of objects to compute the visible scene. Again, the scene description is input to the master process and read-shared among all the slave processes, which compute parts of the scene. Once the parts are completed, the master displays the entire scene. Like

matrix computation, there is very little interference between the slaves except at the block boundaries of the scene buffer.

A parallel version of a printed circuit board inspection program exhibits a very similar data-access pattern, as well as excellent speedup using the read-replication algorithm.

Shortest paths. Similar to the above applications, a large matrix represents the paths between pairs of nodes in a graph. However, elements of this matrix are updated in place as new, better paths are discovered. In our straightforward parallel implementation, the processes make interleaved and mixed read and write accesses to the matrix, with the update rate being high at the beginning and declining as better paths become harder and harder to find. The central-server algorithm is inappropriate because, like the applications above, accesses are frequent.

On the other hand, since access locality is poor, large numbers of block transfers due to thrashing are unavoidable if either the migration or the read-replication algorithm is used. Full replication appears to perform better, especially for the later stages of the computation. Instead of transferring or invalidating a whole block when one entry is updated, only that entry is distributed.

Hypermedia and distributed game playing. Although these two types of applications serve very different purposes, they often share the characteristics of making interactive, concurrent accesses to shared data, with updates to moving, focused regions of the data space as the participants cooperate on or fight over the same areas. The read-replication algorithm may exhibit block thrashing as a result, and full replication again shows its merits.

Distributed locks. Locks are often used in parallel applications for synchronization. Typically, locks require little storage and exhibit poor locality, so that algorithms using block transfers — migration and read replication — are inappropriate. If the demand on a lock is light, a thread will usually find the lock free and may simply lock it, perform the relevant operation, and then unlock it.

Since such operations are relatively infrequent, a simple algorithm such as central server is sufficient. However, if a lock is highly contended for, a process might repeatedly attempt to lock it without suc-

cess, or a "call-back" mechanism might be used to avoid spinning. In either case, the cost of accessing remotely stored locks can be significant, and migrating this lock alone is desirable. Some specialized algorithm seems desirable for distributed locks.

Improving performance

Many variations to the basic distributed shared memory algorithms exist. Many of them improve performance for specific applications by optimizing for particular memory access behaviors, typically by attempting to reduce the amount of communication since costs are dominated by communication costs. Here, we describe two particularly interesting variations and also describe how applications can help improve performance by controlling the actions of the distributed shared memory algorithm. The largest improvements to performance can probably be achieved by relaxing consistency constraints,² something we do not consider here.

Full replication with delayed broadcasts. The full-replication algorithm incurs $S + 2$ packet events on each write operation, where S is the number of participating sites. A slight modification to this algorithm can reduce the number of packet events per write to four, while read operations continue to be executed locally (with no communication overhead).

Instead of broadcasting the data modifications to each site, the sequencer only logs them. A write sent to the sequencer by process P is acknowledged directly and a copy of all modifications that arrived at the sequencer since the previous time P sent a write request is included with the acknowledgment. Thus, the shared memory at a site is updated only when a write occurs at that site. As in the full-replication algorithm, read operations are performed locally without delays.

This variation on the full-replication algorithm still maintains consistency, but has at least two disadvantages. First, it refrains from updating shared memory for as long as possible and therefore does not conform to any real-time behavior. Programs cannot simply busywait, waiting for a variable to be set. Second, it places an additional load on the (central) sequencer, which must maintain a log of data modifications and eventually copy each such modification into a message $S-1$ times.

An optimistic full-replication algorithm. All of the basic algorithms described in the "Basic algorithms" section are *pessimistic* in that they ensure a priori that a process can access data only when the shared data space is and will remain consistent. Here, we describe an algorithm⁷ that is *optimistic* in that it determines a posteriori whether a process has accessed inconsistent data, in which case that process is *rolled back* to a previous consistent state. This algorithm evolved from attempts to increase performance of the full-replication algorithm by coalescing multiple write operations into a single communication packet.

Instead of obtaining a sequence number for each individual write operation, a sequence number is obtained for a series of consecutive writes by a single process. Obviously, this may lead to inconsistent copies of data, since writes are made to the local copy and only later transmitted (in batch) to other sites. If the modified data is not accessed before it reaches a remote site, temporary inconsistencies will not matter. Otherwise, a conflict has occurred, in which case one of the processes will have to roll back.

To roll back a process, all accesses to shared data are logged. To maintain the manageability of the size of these logs (and the operations on them efficient), it is convenient to organize the data accesses into transactions, where each transaction consists of any number of read and write operations bracketed by a `begin_transaction` and an `end_transaction`.

When `end_transaction` is executed, a unique gap-free sequence number for the transaction is requested from a central sequencer. This sequence number determines the order in which transactions are to be committed. A transaction T with sequence number n is aborted if any concurrently executed transaction with a sequence number smaller than n has modified any of the data that transaction T accessed. Otherwise, the transaction is committed and its modifications to shared data are transmitted to all other sites. These transactions will never have to roll back. The logs of a transaction can then be discarded.

Clearly, the performance of this optimistic algorithm will depend on the access behavior of the application program and the application program's use of transactions. If rollbacks are infrequent, it will easily outperform the basic full-replication algorithm. It will also outperform the read-replication algorithm for those appli-

cations that suffer from thrashing, since the optimistic algorithm compares shared memory accesses at the per-data-item level (as opposed to a per-data-block level). The primary drawback of this algorithm, however, is the fact that the shared memory is no longer transparent to the application because the memory accesses must be organized into transaction and because roll backs must be properly handled by the application.

Application-level control with locking. Locks can be used by the application not only for its synchronization needs, but also to improve the performance of the shared memory algorithms. For example, in the case of the migration and read-replication algorithms, locking data to prevent other sites from accessing that data for a short period of time can reduce thrashing.

In the case of the full-replication algorithm, the communication overhead can be reduced if replica sites only communicate after multiple writes instead of after each write. If a write lock is associated with the data, then once a process has acquired the lock, it is guaranteed that no other process will access that data, allowing it to make multiple modifications to the data and transmitting all modifications made during the time the lock was held in a single message without causing a consistency problem. That is, communication costs are only incurred when data is being unlocked rather than every time a process writes to shared data.

Having the application use locks to improve the performance of the shared memory has a number of disadvantages, however. First, the use of locks needs to be directed towards a particular shared memory algorithm: the shared memory abstraction can no longer be transparent. Second, the application must be aware of the shared data it is accessing and its shared data access patterns. Finally, in the case of those algorithms that migrate data blocks, the application must be aware of the block sizes and the layout of its data in the memory.

Despite the simplifying assumptions made in the performance analyses, the essential characteristics of the four basic algorithms are captured in the models used. The concept of distributed shared memory is appealing because, for many distributed applications, the shared memory paradigm leads to simpler (application) programs than

when data is passed directly using communication primitives. Moreover, with respect to performance, numerous implementations have shown that distributed shared memory can compete with and, in some cases, even outperform data-passing programs.

On the negative side, the performance of the algorithms that implement distributed shared memory are sensitive to the shared memory access behavior of the applications. Hence, as we have shown, no single algorithm for distributed shared memory will be suitable for most applications. The performance-conscious application writer will need to choose an appropriate algorithm for an application after careful analysis or experimentation.

In some cases, he or she will want to use different algorithms (for different data) within a single application. Moreover, because these algorithms are sensitive to the access behavior of the applications, it is possible to improve their performance significantly either by fine-tuning the application's use of the memory or by fine-tuning the shared memory algorithm for the access behavior of the particular application, thus eliminating the advantages of transparent shared memory access. We should also emphasize that distributed shared memory may be entirely unsuitable for some applications.

Further work is still needed to make distributed shared memory as versatile as its data-passing counterparts. For example, the distributed shared memory algorithms we have described are not tolerant of faults. Whenever a host containing the only copy of some data items crashes, critical state is lost. Although the central-server and the full-replication algorithms can be made tolerant of single-host crashes (for example, by using a backup server in the case of the central-server algorithm), it is not clear how to make the migration and read-replication algorithms equally fault tolerant.

Compared to data passing, distributed shared memory does not appear to be as suitable for heterogeneous environments at this time, although several research efforts on this problem are currently under way.^{7,9} Consider, for example, the migration algorithm in an environment consisting of hosts that use different byte orderings and floating-point representations.

When a page is migrated between two hosts of different types, the contents of the page must be converted before it can be accessed by the application. It is not possible for the distributed memory system to

convert the page without knowing the type of the application-level data contained in the page and the actual page layout. This complicates the interface between the memory system and the application.

If noncompatible compilers are used for an application to generate code for the different hosts such that size of the application-level data structures differs from host to host, then conversions on a per-page basis become impossible. For example, an additional problem for numerical applications is that, since the application has no control over how often a block is migrated or converted and since accuracy may be lost on floating-point conversions, the result may become numerically questionable.

For these reasons, we consider distributed shared memory to be a useful paradigm for implementing a large class of distributed applications, but do not expect it to become widely available in the form of a single standardized package, as has been the case for remote procedure calls, for example. Rather, we expect that distributed shared memory will be made available in a number of forms from which the application writer can choose. ■

Acknowledgments

Many thanks go to Tim McInerney, who did most of the work implementing the migration and read-replication algorithms for the Sun and DEC Firefly workstations, and to Orran Krieger, who designed and implemented the optimistic full-replication algorithm. The anonymous reviewers and the editors provided numerous valuable suggestions for improvements.

References

1. R. Bisiani and A. Forin, "Multilanguage Parallel Programming of Heterogeneous Machines, *IEEE Trans. Computers*, Vol. 37, No. 8, Aug. 1988, pp. 930-945.
2. D.R. Cheriton, "Problem-Oriented Shared Memory: A Decentralized Approach to Distributed System Design, *Proc. Sixth Int'l Conf. Distributed Computing Systems*, May 1986, pp. 190-197.
3. K. Li, "Shared Virtual Memory on Loosely Coupled Multiprocessors," PhD thesis, Dept. of Computer Science, Yale Univ., 1986.

4. K. Li and P. Hudak, "Memory Coherence in Shared Virtual Memory Systems," *ACM Trans. Computing Systems*, Vol. 7, No. 4, Nov. 1989, pp. 321-359.
5. A. Forin et al., "The Shared Memory Server," *Proc. 1989 Winter Usenix Conf.*, Jan. 1989, pp. 229-243.
6. R.E. Kessler and M. Livny, "An Analysis of Distributed Shared Memory Algorithms," *Proc. 9th Int'l. Conf. Distributed Computing Systems*, CS Press, Los Alamitos, Calif., Order No. 1953, June 1989, pp. 498-505.
7. O. Krieger and M. Stumm, "An Optimistic Algorithm for Consistent Replicated Shared Data," *Proc.-1990 Hawaii Int'l Conf. on System Sciences*, CS Press, Los Alamitos, Calif., Order No. 2009, Vol. 2, 1990, pp. 367-375.
8. B.D. Fleisch and G.J. Popek, "Mirage: A Coherent Distributed Shared Memory Design," *Proc. 12th ACM Symp. Operating System Principles*, Dec. 1989, pp. 121-222.
9. S. Zhou, M. Stumm, and T. McInerney, "Extending Distributed Shared Memory to Heterogeneous Environments," *Proc. 10th Int'l. Conf. Distributed Computing Systems*, CS Press, Los Alamitos, Calif., Order No. 2048, May-June 1990.
10. S. Ahuja, N. Carriero, and D. Gelernter,

"Linda and Friends," *Computer*, Vol. 19, No. 8, Aug. 1986, pp. 26-34.

11. L. Lamport, "How to Make a Multiprocessor Computer that Correctly Executes Multiprocess Programs," *IEEE Trans. Computers*, Vol. 28, No. 9, Sept. 1979, pp. 690-691.
12. M. Dubois, C. Scheurich, and F.A. Briggs, "Synchronization, Coherence, and Event Ordering in Multiprocessors," *Computer*, Vol. 21, No. 2, Feb. 1988, pp. 9-21.
13. J. Archibald and J.L. Baer, "Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model," *ACM Trans. Computing Systems*, Vol. 4, No. 4, Nov. 1986, pp. 273-298.

His research interests are in the area of computer systems.

Stumm received a diploma in mathematics and a PhD in computer science from the University of Zurich in 1980 and 1984, respectively. He is a member of the IEEE Computer Society and ACM.



Songnian Zhou is an assistant professor in the departments of computer science and electrical engineering at the University of Toronto. His research interests include operating system design, distributed and parallel systems, computer networks, and performance evaluation.

Zhou received the BS degree from Northeastern University, Boston, in 1982, and MS and PhD degrees in computer science from the University of California, Berkeley, in 1984 and 1987, respectively. He is a member of the IEEE Computer Society and ACM.



Michael Stumm is an assistant professor in the departments of electrical engineering and computer science at the University of Toronto.

The authors can be contacted at the University of Toronto, Toronto, Canada, M5S 1A4.



**CALL
FOR
PAPERS**

The European Design Automation Conference

Amsterdam, The Netherlands, 25-28 February 1991

EDAC-90 attracted in Glasgow 450 delegates and 159 attended the 3 tutorials. From more than 300 submitted papers, 120 verbal presentations and 30 poster presentations were organized.

EDAC-91 will be held in Amsterdam. The Conference will cover all areas of the design process of electronic circuits and systems from concept to manufacture and includes CAD and DA tools for analog, digital, VLSI, microwave and high-speed electronics.

In addition, a panel session, a number of fringe meetings, a day of tutorials and vendors presence are planned.

EDAC Conferences are run by EDAC, a non-profit association, in association with IEEE/DATC, ACM/SIGDA and other European technical societies.

The deadline for receipt for papers is 3 September 90 and intending authors should contact the Secretariat for instructions for submission.

For further details, contact:

Secretariat, EDAC 91
CEP Consultants Ltd.
26-28 Albany Street
Edinburgh EH1 3QH, SCOTLAND
Telephone: +44 31 557 2478
FAX: +44 31 557 5749.



THE INSTITUTE OF ELECTRICAL AND
ELECTRONICS ENGINEERS, INC.

Distributed Hierarchical Control for Parallel Processing

Dror G. Feitelson and Larry Rudolph
Hebrew University of Jerusalem

The development of operating systems for parallel computers has closely followed that for serial computers. At first, the most advanced parallel computers ran in batch mode or single-user mode. At best, they allowed a static partitioning among a number of users. They were typically designed with a specific computational task in mind or for a certain class of computations. Like serial computers, they are currently evolving towards general-purpose, interactive, multiuser parallel systems.

To explain the underlying motivation for our work, we note that a general-purpose, interactive, multiuser, multiprogramming parallel environment has the following advantages (in addition to the traditional advantages in uniprocessor environments, such as cost effectiveness):

- This environment provides users with a spectrum of computational powers, covering the range from personal computers to supercomputers. A user requiring more computational power can simply use more processors. Thus, a short response time for both simple and computationally intensive tasks is possible.

- The spectrum of powers also aids program development and evaluation. Initially, only one processor is needed. Additional processors can be added later with-

A novel design using a hierarchy of controllers effectively controls a multiuser, multiprogrammed parallel system. Such a structure allows dynamic repartitioning according to changing job requirements.

out changing the interface. Contrast this with network-based systems, in which program development takes place on a front-end processor and the program is downloaded to the parallel machine for execution.

- This environment makes efficient use of hardware by sharing it between several jobs with complementary requirements at every given moment. This generalizes common practice on uniprocessors, such

as overlapping the computation of one program with the I/O of another.

- This environment can aid in the development of large, modular parallel systems by allowing the different modules to be programmed and executed as if they were independent jobs. This relieves the program developer of the need to fit modules together and to coordinate the division of resources among them.

Support for the simultaneous execution of a number of parallel programs implies that the system must coordinate the partitioning of resources among the running programs, as well as any dynamic repartitioning required due to the changing needs of executing programs. Of prime importance is the allocation of processors to the different programs. Two basic driving forces govern this. On the one hand is the drive for efficiency, which usually implies that the various devices all be kept busy. On the other hand is the drive to be fair to all users, giving them equal shares of the available resources (or unequal shares proportional to their needs). Achieving fairness is complicated by the fact that different users have different types of requirements, sometimes hard to compare. For example, while some users execute computation-bound programs, others run I/O-bound ones. These two forces, effi-

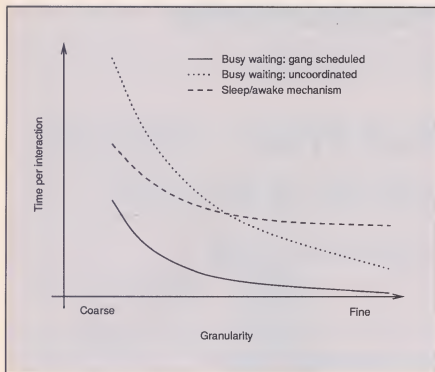


Figure 1. Busy waiting is the most efficient way to implement fine-grained interactions between threads, provided they are gang scheduled. As the size of blocks between synchronization points becomes smaller, the time to execute them tends to zero. With a sleep/awake mechanism, the interaction time is bounded from below by the context-switch overhead.

ciency and fairness, sometimes conflict. We cannot prevent programs that cause inefficiencies from running, as this violates fairness.

Note that it is also desirable for general-purpose parallel systems to support various abstractions. Support for abstractions is, after all, an important duty of operating systems. As in the case of the other important duty, resource allocation, the abstractions supported by a parallel system necessarily differ from those on a uniprocessor. For example, the conventional process abstraction might be replaced by two types of processes: heavy-weight processes (similar to the well-known Unix process) and light-weight processes (sometimes called *threads*).

Heavy-weight processes allow modular and structured design of large systems by creating distinct contexts for independent computations, separated and protected from each other. This is also true for independent users. Threads allow fine-grained parallelism; many threads can exist within the context of one process, cooperating to perform the computation and sharing address space, open files, etc. Likewise,

abstractions for information interchange should be augmented with more powerful mechanisms than files. Various flavors of message passing fall into this category.

This article focuses on the coordination of activities in a parallel machine rather than on the abstractions presented to the user. The discussion emphasizes the mapping and scheduling of processes on processors. We start by outlining the specific goals that the system must meet in this area, then present the distributed hierarchical control concept designed to meet these goals. To keep matters simple, we limit the discussion to the shared memory model with uniform memory access. (However, each processor may also have a private local memory in addition to the uniform-access shared memory.) The results extend to other models.

We give detailed algorithms for the new control structure, as well as simulation results that tabulate its effectiveness. Comparisons to other systems that attempt to solve the same problems in different ways provide an overview of approaches to various issues in parallel operating systems.

Design goals

One goal of an operating system is to allow the user to program at a higher level of abstraction. Unfortunately, in many parallel systems the user program largely dictates the allocation of resources. The Occam programming language, for example, requires the user to explicitly assign processes to processors and virtual communication channels to physical links, leaving only the runtime implementation to the operating system. Past experience with uniprocessors, however, suggests the benefits of keeping the level of user involvement as low as possible. Partitioning the job into threads is part of program development and, as such, is the responsibility of the user or compiler. Mapping and scheduling these threads is the responsibility of the operating system and should be out of bounds to users. Likewise, we should avoid user involvement in memory management.

The main argument for allowing user intervention in parallel systems' operating system tasks revolves around the importance of efficiency and speed on these systems and the belief that only the user can understand the application well enough to decide on the optimal system parameters. The validity of this argument depends on the number and complexity of the parameters, since it can be extremely difficult to produce an optimal configuration in the face of too many details. We therefore argue that, to make parallel programming easier, high-level principles should replace detailed user knowledge, thus restoring the traditional distinction between the responsibilities of the user and the operating system. Specifically, we propose the following guidelines for the design of an operating system to support dynamic resource partitioning of a parallel computer among a number of users, without help from the users:

- *Spatial locality.* The principle of locality in the logical structure of a program — especially in its address space — is well known from uniprocessors. In massively parallel systems this concept extends to the physical structure of the machine, since we can expect nonuniform distances between different elements. The principle of spatial locality indicates that we should map strongly interacting entities in close proximity, because interactions across large distances cost more. In other words, the system must endeavor to match the logical locality with the physical locality. For

example, threads created by a single construct should be mapped to a cluster of tightly coupled processors. Also, their address space should be mapped to memory modules within the same cluster. While largely irrelevant to uniform-memory-access machines, this principle is an important concept when considering extensions to nonuniform-memory-access machines.

• *Simultaneous actions.* The most prominent difference between parallel machines and uniprocessors is that in a parallel machine many system activities relating to the same program can happen concurrently. It is often important that a set of these activities happen simultaneously. For example, the concept of context switching from uniprocessors can be extended to *multithreaded switching*, that is, the simultaneous switching (across a number of processors) from the threads of one task to the threads of another task. We need this to implement *gang scheduling*, which we define as the scheduling of a group of threads to run on a set of processors at the same time, on a one-to-one basis. Gang scheduling helps support the user's intuitive model of a totally dedicated parallel machine and allows the efficient implementation of fine-grain parallelism (see Figure 1).

• *Balancing.* An important principle importable directly from uniprocessor systems is that of balancing. Adequate balancing ensures that the whole system works at full capacity. This includes balancing the workload on different parts of the system, for example, by scheduling a good mix of computation-bound and I/O-bound jobs, and balancing the workload on replicated instances of a certain device, such as by interleaving an address space on a number of memory modules. This principle becomes even more important in multiprocessor systems, where all devices, including the processor itself, are replicated.

The implementation of a parallel operating system that coordinates the use of shared resources according to the above guidelines must be carefully designed to avoid compromising system scalability. A system that needs to consult detailed constituent data before making a decision would become a bottleneck. The quest for a truly scalable system that can handle hundreds or even thousands of independent processors underlies the distributed hierarchical style of the design presented in the next sections.

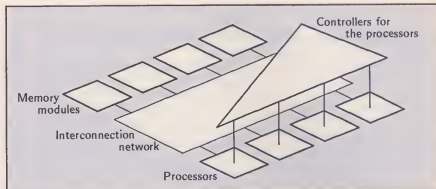


Figure 2. Schematic representation of the logical structure of a parallel computer (bottom), with a hierarchical control structure over the processors. The control hierarchy is orthogonal to the architecture of the machine.

Distributed hierarchical control

The principles of locality, simultaneous actions, and balancing can help guide the activities of a general-purpose, multiuser parallel system. A novel control structure is proposed for the operating system, referred to as *distributed hierarchical control*. To appreciate the ways in which this control structure departs from common practices, let us first examine alternative approaches to the control of parallel computers.

Configuration and control. Generally speaking, the control of resources in a parallel computer can be centralized or distributed, or some combination of the two. Specifically, the following configurations have been proposed (see Hwang and Briggs,¹ section 7.4):

(1) *Master-slave configuration.* In this centralized approach, the operating system runs on the master processor and downloads work to the slaves. The master is a critical component in such systems; if it fails or becomes overloaded, the whole system is affected. As a consequence, this design has limited scalability.

(2) *Distributed replicated system.* In this decentralized approach, the operating system runs on each processor with a full copy of its data structures. This avoids the critical master processor but places a large demand on storage space. Also, the main point in parallel computing is that the processors interact easily, and such interaction is difficult to control in a loosely coupled system.

(3) *Symmetrical structure with shared*

data. Many systems (such as Symunix on the New York University Ultracomputer, Xylem on Cedar, and multiprocessor implementations of Mach) aspire to this ideal. The processors play symmetrical roles, are tightly coupled, and share their data structures. This allows better load balancing across the system but requires careful coding so as not to compromise the integrity of the system. Special primitives are required to avoid inefficiencies resulting from conflicts in the access of shared data. In addition, coordinating the activities of multiple processors (such as gang scheduling of related threads) is nontrivial.

The distributed hierarchical control concept departs from this latter configuration, being inherently asymmetrical. In fact, it more closely resembles a master-slave configuration than a symmetrical system. However, instead of a uniprocessor master controlling a set of parallel slaves, the master is itself a parallel machine. We refer to the master's processors as *controllers*. A tree of controllers supervises the activities of the parallel slaves (see Figure 2). Controllers in higher levels of the tree care for activities that involve a large degree of parallelism, with the root responsible for activities that need to coordinate all the slaves in a concentrated effort. Note that the root controller does not limit performance; it does not become a sequential bottleneck. The root controller participates only in the scheduling of very large groups; due to their size, only one such group can be executed at a time. The root does not mediate between its subordinate controllers; that would indeed cause a bottleneck.

Controllers in higher levels do not keep track of all the details relating to the indi-

Hierarchical control and gang scheduling

A small number of projects over the last dozen years used the idea of a hierarchical control structure for a parallel computer. The EGPA (Erlangen General-Purpose Array) project¹ was based on a pyramidal hardware hierarchy with a function similar to our tree of controllers. The processors on the bottom layer were connected in a square mesh pattern and executed user code, while higher levels were used by the operating system. However, it was never extended beyond two levels, and no operating system algorithms were published for it. A hardware hierarchy for load balancing was proposed in the AMPS (Applicative Multiprocessing System) project,² but was later dropped.

Systems like Micros³ and Chopp⁴ (Columbia Homogeneous Parallel Processor) used the software approach, creating a virtual hierarchy dynamically as needed. Thus, some of the processors were used for control, and those left over were available for users. These hierarchies were only used to map groups of processes to processors, thus partitioning the machine. Once a group of processes was mapped, it was executed without preemption until all the processes terminated. Therefore, these systems were not interactive.

The same is true of Pasm (Partitionable SIMD/MIMD),⁵ which uses a static two-level hardware structure. The mapping algorithms developed for Pasm are similar to ours in their dealing with fragmentation issues, but they are based on a centralized mapper with full knowledge. Hence, they are less scalable. Because only two levels are used, the partitioning is also less flexible. Xylem, the Cedar operating system,⁶ also employs a two-level scheme that uses clusters of processors as the unit of allocated processing power. All these systems support batch-style gang scheduling.

The only interactive parallel operating system so far to support gang scheduling of groups with varying sizes was Medusa, one of the two operating systems for Cm⁷. Actually, this system introduced the concept. It used a very simple scheduling algorithm, where a central controller attempts to pack groups into sets so that the sum of the sizes of all the groups in a set is smaller than the number of processors. Then the controller broadcasts the ID of the set that should be scheduled next. This scheme was sufficient because of the size of Cm⁷ (50 processors) and the relatively low load placed on the system.

References

1. W. Händler et al., "A Tightly Coupled and Hierarchical Multiprocessor Architecture," *Computer Physics Comm.*, Vol. 37, Nos. 1-3, July 1985, pp. 87-93.
2. R.M. Keller, G. Lindstrom, and S. Patil, "A Loosely Coupled Applicative Multiprocessing System," *AFIPS Conf. Proc.*, Vol. 48, June 1979, pp. 613-622.
3. A.M. van Tilborg and L.D. Wittie, "Wave Scheduling — Decentralized Scheduling of Task Forces in Multicomputers," *IEEE Trans. Computers*, Vol. C-33, No. 9, Sept. 1984, pp. 835-844.
4. H. Sullivan, T.R. Bashkow, and D. Klappholz, "A Large-Scale, Homogeneous, Fully Distributed Parallel Machine, II," *Proc. 4th Ann. Int'l Symp. Computer Architecture*, Mar. 1977, pp. 118-124.
5. D.L. Tuomikoski and H.J. Siegel, "Task Scheduling on the PASM Parallel Processing System," *IEEE Trans. Software Eng.*, Vol. SE-11, No. 2, Feb. 1985, pp. 145-157.
6. P. Emrath, "Xylem: An Operating System for the Cedar Multiprocessor," *IEEE Software*, Vol. 2, No. 4, July 1985, pp. 30-37.
7. J.K. Ousterhout, *Medusa: A Distributed Operating System*, University Microfilms International Research Press, Ann Arbor, Mich., 1981.

vidual slaves' states; rather, they use condensed data supplied by controllers in lower levels. When global coordination is not needed, the controllers in the lower levels control the slaves according to local considerations. Controllers in the same

level of the hierarchy control disjoint sets of slaves, thus providing a spatial partitioning of the machine.

This structure provides local control when possible and global coordination when needed. Consider the interesting

analogy of modern democratic governments. The higher levels of government delegate much of their authority to lower, local officials, while they themselves deal only with global issues. The local officials supply condensed and abstracted data about the various locales, thus relieving the top levels from the flood of details. Nobody knows everything going on in the system, but it seems to work.

The idea of distributed hierarchical control should be distinguished from other hierarchical or clustered approaches proposed for parallel computing. Such proposals are usually motivated by memory latency considerations and the desire to build a scalable system (examples include Cm⁸ and Cedar). The hierarchy is therefore visible to the user in its effect on the communication between various processors. With distributed hierarchical control, the operating system uses the hierarchy as a control structure. In principle, it is orthogonal to the interconnection pattern between the processors as the user sees it. The sidebar "Hierarchical control and gang scheduling" reviews several related proposals that also use a hierarchical structure for control.

Implementation issues. Implementation of a distributed hierarchical control scheme can take one of two forms: a software abstraction of a separate hardware structure.

At first glance, the software abstraction method seems more natural — the controllers are simply mapped onto the underlying processors, which switch between user and supervisor modes of operation. This approach has a number of drawbacks, however. For one thing, it wastes the power of a tightly coupled machine on operating system tasks that do not require such close interactions. In addition, the switching between modes might interfere with the gang scheduling of related user threads.

Users of uniprocessor systems accept the fact that the operating system uses up to 25 or even 50 percent of the CPU cycles. Why not devote a similar percentage of the processors to the operating system in a multiprocessor system? The construction of a control hierarchy in hardware might seem wasteful because at least some of these processors will lie idle at any given moment. However, such a construction has the advantage of allowing the architect to optimize distinct processors for their intended tasks. Specifically, we could make the controllers in the operating system hierarchy less powerful than the processes

sors provided for users; for example, these processors do not need floating-point accelerators. In addition, they need only a simple interconnection pattern, no shared memory, and a small amount of local memory.

While such an approach introduces some asymmetry and might increase the complexity of the system, this increase is small—all the controllers in the hierarchy are uniform. Likewise, uniprocessor architectures also suffer from hardware complexity added to support operating system functions such as virtual memory.

The processor-chauvinistic approach that laments any idle time of the operating system controllers is out of place for two reasons. First, once processors have a specific task, it is wrong to think that their idle time reflects a degradation of system performance. Such an approach would imply that we should not use integer-based programs when floating-point accelerators are available, and that we should not execute computation-bound jobs on machines with I/O processors. Second, the correct approach should be "expensive-part chauvinism," meaning we should identify the most expensive part of the machine and keep it busy at the possible expense of other, less expensive parts.

Thirty years ago the most expensive part was the processor. In the parallel machines of today and the near future, the expensive part is the tightly coupled structure of processors, interconnection network, and memory modules. Thus, we should keep this structure busy executing user code by sharing it among a number of user programs. System-related activities should be offloaded from this structure and executed by less expensive peripheral processors. Useful computation can overlap with operating system work, just as it traditionally overlaps with I/O.

In fact, using separate processors for operating system tasks is not new at all. It exists not only in certain master-slave configurations, but also in the majority of supercomputers, where a host takes care of most of the scheduling and I/O operations. The innovation in the present proposal lies in the fact that the operating system also runs on a parallel machine tightly coupled with the target machine.

Just as operating system processing is offloaded from the user processors, so operating system storage should be offloaded from the expensive shared memory. With distributed hierarchical control, some of the system tables and the

kernel code move to the controllers' local memories.

Control over processors

The fact that a multiprocessor has many processors adds a new dimension to the problem of allocating processing power to user programs. The operating system must decide not only which program executes when, but also where. Specifically, we can allocate groups of processors to distinct parallel programs that run side by side, rather than always giving the whole machine to one program and switching from one program to another. To ensure efficient and fair use of the processors, this partitioning should change dynamically over time. Thus, it seems that the processor allocation problem on a multiprocessor more closely resembles the memory allocation problem on a uniprocessor than the scheduling problem on a uniprocessor. Indeed, the distributed hierarchical control scheme calls to mind the "buddy system" method for memory allocation.²

The analogy between the processor allocation problem and the memory allocation problem extends to the adverse effects suffered by various algorithms for their solution.³ *Processor fragmentation* refers to a situation in which some processors are left over when others are allocated, and the leftover processors are either insufficient in number or unsuitably organized to support the requirements of waiting jobs.

The term *processor thrashing* describes a situation in which threads cannot make any progress because related threads with which they must interact do not run at the same time, resulting in excessive context switching. Gang scheduling is meant to eliminate this phenomenon, analogous to allocating memory frames for a whole working set of pages at once to forestall additional page faults.

There have been attempts to extend the analogy even further. The main idea is to provide the user with *virtual processors* (analogous to virtual memory pages), leaving hidden the mapping to real processors. A full implementation of this extension suffers from a number of drawbacks:

- The success of virtual memory is a result of the locality property of most programs; at any given time, these programs use only a small part of their virtual address space. There is no evidence that a parallel program would use only a small

portion of its virtual processors at any given time.

- Virtual processors encourage the user to request processors without regard for hardware limitations, which might cause inefficiencies in the mapping. For example, a user might request many more processors than are physically available.

- Providing a virtual processor abstraction might be inefficient, as it adds another software layer that is not strictly necessary.

- The operating system should map virtual processors to physical ones subject to interaction patterns between the virtual processors, yet typically we do not know these patterns in advance. For example, we might obtain large benefits by mapping two virtual processors that use the same data to the same physical processor, thereby allowing them to share its cache.

Note, however, that any high-level programming construct implies virtual processors in some sense. The point is that they should not be too virtual, but rather have a strong relation to the real machine. A gang-scheduling policy does this by allowing the real processors to relate to each other as do real processors.

Approaches to the mapping and scheduling problems. The problem of mapping processes to processors and the problem of scheduling these processes to run are not necessarily independent. In fact, we can consider them together as a variant of the two-dimensional bin-packing problem, where one dimension represents the processors and the other represents time. Each group of threads is then represented by a rectangle that defines how many processors it needs for how long. This depends on the assumption that the number of threads in a group does not change and, specifically, that they all terminate at the same time. However, the assumption does not imply a constant number of threads in the whole program. Existing groups can terminate and new groups can be created dynamically.

Existing algorithms for mapping and scheduling are not limited to the packing of predefined rectangles. Four approaches stand out in the literature:

- (1) *Self-service.* This approach contains no mapping per se. It maintains a central queue of ready threads, and idle processors in search of work simply help themselves. This is sometimes called *load sharing* to distinguish it from load-balancing

- Preemption is used to support an interactive system with many independent jobs, as opposed to batch scheduling.
- Related threads are gang scheduled one-to-one on a set of processors to ensure efficiency by matching the user's concept of a parallel machine.
- Loads are balanced (to the degree possible with limited knowledge) to help maintain fairness.
- Threads are not moved once started, to keep contexts local and to increase caching efficiency.
- Centralized control that might become a bottleneck is not used; rather, a hierarchical structure provides coordination when needed without compromising scalability.

Figure 3. Characterization of the distributed hierarchical control scheme.

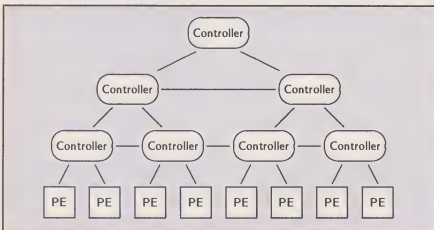


Figure 4. Schematic representation of the control hierarchy over the processors. (PE: processing element.)

schemes in which work is allocated on a more permanent basis.

This approach has certain advantages. It distributes the load evenly across the processors, assures that no processors lie idle while work remains, and uses no centralized control. Its disadvantages include

- The central queue might become a bottleneck when many processors look for work at the same time. We need special hardware primitives such as fetch-and-add to avoid this danger.⁴
- Preempted threads are unlikely to resume execution on the same processor. Thus, we lose the opportunity to exploit any storage associated with the processor for local state information (such as the stack). Caching also becomes less efficient.
- It appears that coordinating the processors to guarantee gang scheduling of related threads requires a high overhead. This overhead is unacceptable unless we adopt a batch-scheduling discipline.

Despite these objections, this scheme is

widely used in contemporary parallel machines. Examples include Symunix on the NYU Ultracomputer, Dynix on the Sequent Balance, and Umax and Mach on the Encore Multimax. (For descriptions of most of the systems mentioned here and in the following paragraphs, refer to the book by Almasi and Gottlieb.⁵)

(2) *Nonpreemptive mapping.* The opposite of the self-service approach, this approach features implicit scheduling defined by the mapping. The processors are divided between the running programs in a quasi-static manner. Each program is allocated a number of processors equal to the number of threads in the program, for the duration of the program execution. When the program terminates, the processors return to the general pool for possible allocation to another program.

This scheme assures gang scheduling and efficient caching (since threads do not move). But it does not suit interactive systems, it cannot deal with programs with more threads than processors, and processors stay idle if queued jobs require more

processors than are currently available.

The Pasm, Micros, and Chopp projects use nonpreemptive mapping.

(3) *Local queues.* This approach is typically used in distributed systems and multicomputers rather than in multiprocessors. Examples include transputer-based systems and hypercubes. It involves the mapping of threads to processors, whereupon each processor employs time sharing to service the threads mapped to it.

This scheme suits a loosely coupled system. Because threads stay in the same processor, caching is efficient and context information does not have to be moved. The main disadvantage is that loads might become skewed, compromising fair scheduling and degrading performance. In extreme cases certain processors may lie idle while others become overloaded. This implies that a runtime load-balancing mechanism should be introduced.⁶ Another disadvantage—the lack of coordination between processors—makes gang scheduling difficult to achieve.

(4) *Precedence graph.* The precedence graph shows how the program is decomposed into independent basic blocks (the nodes of the graph) and how these basic blocks depend on each other (directed arcs). The execution time of each block is usually also given. The mapper analyzes this graph and decides which blocks should be executed when and on what processor.

This scheme typically relies on a central mapper using rather sophisticated algorithms (see, for example, Lo⁷). This is considered permissible because the mapper is only invoked once to determine the scheduling sequence of the threads in a new job. Such algorithms are used only in real-time systems.

The drawbacks are that the algorithms require too much knowledge: first, the generation of the precedence graph, then the expected runtimes of the basic blocks to yield the total runtime of the resulting schedule for evaluation. In addition, the algorithms cannot coordinate the scheduling of independent jobs.

The Cedar multiprocessor system also uses a precedence graph, but only to ensure correctness, not to minimize response time. Therefore, it does not really belong in this class.

The distributed hierarchical control solution to mapping and scheduling attempts to achieve the advantages of the different schemes while avoiding the disadvantages. The guidelines that characterize this solution appear in Figure 3. The algorithms are

designed for a hierarchical structure of controllers interconnected to form a binary X-tree, as in Figure 4. We chose the binary tree for the purpose of presentation only, as it simplifies the algorithms. A real implementation would probably use a tree of higher degree, thus reducing the height of the tree and the number of controllers.

To keep the algorithms straightforward, we make two additional simplifying assumptions. First, we assume that all the threads in the system are always ready to run. We do not have to keep track of threads that become blocked and unblocked. While somewhat unrealistic, this assumption is necessary because the whole issue of how to treat blocked threads in conjunction with a gang-scheduling requirement is far from being resolved. Once the desired semantics are decided, the algorithms can be refined to support them. Second, we assume that the architecture supports a uniform-access shared memory, and therefore the mapping of threads to processors is independent of the allocation of memory. In a nonuniform memory access machine, we must take the available memory into account when performing the mapping.

Mapping and load balancing with distributed hierarchical control

A good mapping algorithm should pave the way for efficient gang scheduling. The requirement of gang scheduling indicates that threads should be mapped to distinct processors. To fulfill this requirement, related threads must first be identified as such. We refer to a set of related threads as a *group*. A possible way to identify groups is to use the syntactic structure of the program as an indicator for the degree of proximity between threads.³ In other words, we assume that threads generated by the same construct interact strongly, while we hope that threads from distinct constructs do not. In terms of the process tree generated by the task, this means that at any given moment the sets of leaves with a common parent constitute the groups of threads that should be mapped together and gang scheduled (see Figure 5). This notion is also well suited to the dynamics of program execution, because the threads in such groups are actually spawned together.

The mapping of a newly spawned group of threads proceeds as follows: The group

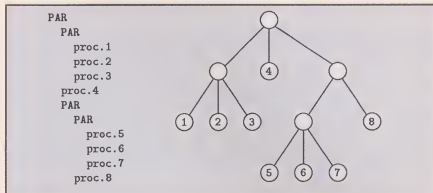


Figure 5. A code segment (using Occam notation) and its process tree. Sets of leaf processes with a common parent are assumed to interact strongly with each other and are therefore gang scheduled. This case has two such sets: { 1, 2, 3 } and { 5, 6, 7 }.

In controllers:

```
total = total[left-child] + total[right-child]
groups = max{ groups[left-child], groups[right-child] }
+ number of groups controlled by this controller
threads = threads[left-child] + threads[right-child]
+ number of threads in groups controlled by this controller
```

In leaf processors:

```
total = number of threads mapped to this processor
groups = number of single threads on this processor
threads = number of single threads on this processor
```

Figure 6. The generation of static information in the tree of controllers. Single threads are threads that do not belong to any group. This information is later used for load balancing, mapping, and scheduling.

is spawned by an existing thread running on some processor. The request to create the new group is passed from the processor to its direct controller, and then up the tree of controllers until it reaches the level that caters to groups of this size. (Counting from the leaves upward, level i manages groups in the range of sizes from $2^{i-1} + 1$ to 2^i .) The request is then passed among controllers within that level to the least loaded, most quickly found controller — this is the reason for the horizontal connections in the X-tree of controllers. Finally, the group is mapped to the processors under the chosen controller.

This procedure divides the events leading to the execution of the new threads into two steps: load balancing between the controllers and mapping to the processors under a specific controller. The load-balancing step requires controllers to have

some notion of the load under them. Because of scalability constraints, this cannot be a detailed knowledge of the load on each of the underlying processors. Therefore, the details are condensed into a single value — the total number of threads on all the processors — as they are passed up the tree. This data (and other data used for scheduling) is maintained recursively as shown in Figure 6.

This measure of load can be used to select the best controller. However, the scalability requirement precludes comparing the loads of all controllers in the level to find the optimal one. Implementation is further simplified if each controller connects to only a few other controllers and data transfers only between neighbors. In this case we can envision load balancing as a diffusion of load along the edges of a network of controllers. The effectiveness

Simulation methodology

The system we have presented here is a paper design, not yet implemented. We therefore cannot provide measured performance figures. There are two ways to estimate the performance of a system without implementing it: one, by analytical modeling, and two, by simulation. We chose simulations because it is possible to simulate the system under study in a direct manner, thus lending credibility to the results. Analytical modeling typically requires additional assumptions, and such assumptions might have unforeseeable influences on the results.

We used special checks to verify the correctness of the simulations. For example, the running time of groups of different sizes and the total processor idle time were tabulated independently. At the end of the simulation, the sum of the runtimes weighted by the number of groups of each size, plus the idle time, should equal the total processing resources.

The results presented are derived from the simplest possible simulations. We simulated systems of 16, 32, and 64 processors. The workloads were synthetic; each consisted of a number of groups with sizes selected at random from a uniform distribution. The sizes typically ranged from 1 to p , the number of processors. In the simulations used to check the load-balancing algorithms, we also used a range of 1 to $p/10$. This was necessary to bring out the differences between the various load-balancing schemes, because large groups always need the whole machine and therefore mask the effects of load balancing. We set the number of groups so that the total number of threads in all the groups would conform with the load parameter, which was an input parameter of the simulation. Each point in the graphs represents an average of up to a thousand such workloads.

References

D.G. Feltelson and L. Rudolph, "Mapping and Scheduling in a Shared Parallel Environment Using Distributed Hierarchical Control," to appear in *Proc. 19th Int'l Conf. on Parallel Processing*, Aug. 13-17, 1990, St. Charles, Ill.

of such diffusion dynamics depends on a number of parameters:

- The interconnection degree and topology.
- The propagation distance allowed. Loads can move only from a controller to its direct neighbor, or to the neighbor's neighbor, etc.
- The driving force, meaning the difference in loads or some other function of the loads. (See, for example, Lin and Keller.⁹)

The effectiveness of load-balancing depends heavily on workload statistics. Simulation results using a workload of small groups indicate that fairly simple schemes can provide very good performance. (For a short discussion of the simulation methodology, see the sidebar on that topic.) The controllers in the simulations were connected in a linear array or in a hypercube pattern. A request could move a distance of either one or three hops.

Figure 7 shows the distributions of loads generated by the different schemes. It indicates the near optimality of linear connections with three hops and cube connections with only one hop. Linear connections with only one hop do not provide enough bandwidth for good balancing, but even this is much better than no balancing at all.

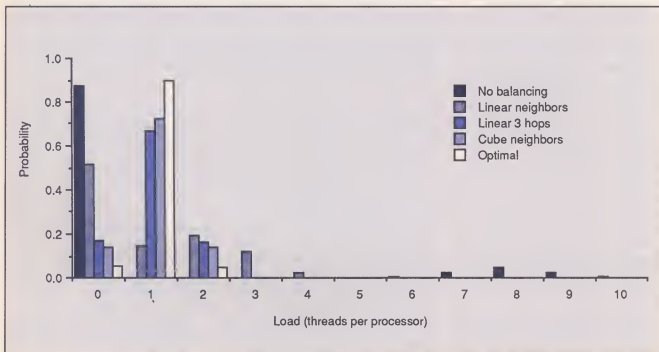


Figure 7. Simulation results for the distribution of loads generated by various load-balancing schemes, for 32 processors under a low load of small groups only. For each scheme the bars show the probability that a certain processor would have a specific load.

When the workload contains large groups, the results are completely different. Because large groups need more processors to support gang scheduling, the load is always spread across the machine. Therefore, overloading of some processors while others lie idle is unlikely.

Simulations for a load with uniformly distributed group sizes from 1 to p , the number of processors, support these observations. Indeed, the results indicate distributions that include large groups do not require active load balancing. These results highlight an interesting difference between gang scheduling systems, where load balancing results automatically because of the gang scheduling, and distributed systems, where load balancing must be handled explicitly.⁶

Balancing the loads on different processors is important because it reduces the variance in the runtimes given to distinct threads and thus improves the fairness. But the balancing must not cause substantial fragmentation of the processors, because such fragmentation makes it harder to map independent jobs side by side. Having chosen the least loaded controller (under the restrictions of the load-balancing scheme), the new threads have to be mapped onto processors under its control. This is done according to the minimal-fragmentation criterion cited above. A controller at level i that has to map a group of s threads divides it into two disjoint subgroups of sizes s_1 and s_2 , one for each of its subordinate controllers. The division is such that one of the groups has exactly 2^{i-1} threads in it, and the other has the rest of the threads. Because the capacity of each subordinate is exactly 2^{i-1} , such a group does not cause fragmentation on the controller that receives it.

Support for gang scheduling

We can define the scheduling problem as the problem of choosing which job to execute. In an interactive system, the question of how long to execute it compounds the problem. Both of these questions become even more complicated in a multiprocessor system, where a number of jobs might be scheduled to run simultaneously on disjoint sets of processors.

The distributed hierarchical control scheduling algorithm is based on the same hierarchy of controllers used for the mapping. Not surprisingly, each controller is directly responsible for scheduling the

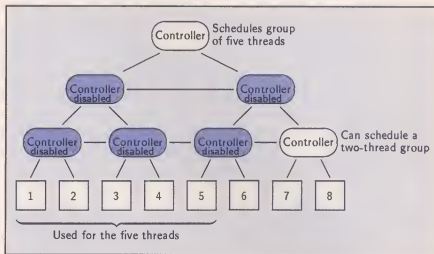


Figure 8. Selective disabling can increase efficiency by allowing leftover processors to schedule smaller groups.

threads of groups mapped to it. The scheduling proceeds as follows: At any given moment a front of controllers across the tree actively schedules groups of threads on the underlying processors. In each scheduling round this front starts at the root and sweeps down the tree. Controllers above the front have completed their scheduling for the current round and lie inactive, waiting for the next round. Controllers below the front are disabled by the currently active controllers in the front.

The front is created on a local basis by the following interactions between adjacent controllers: Each controller (starting with the root) divides its time into two phases. In the first phase it disables its subordinate controllers and directs the underlying processors to schedule the threads of its own groups. To do so, the active controller sends a wave of interrupts down the subtree rooted at its location. It sends a new wave for every group scheduled. Because the wave reaches the processors at the leaves at about the same time, it effectively gang schedules all the threads in a group. After a controller completes the scheduling of its groups, it enters the second phase. In the second phase it reactivates its child controllers so that they, too, can schedule their groups, thus moving the active front downwards. Naturally, a controller only executes this algorithm when not itself disabled by its superiors.

An obvious optimization of the basic algorithm is not to disable all subordinate controllers upon scheduling a group of threads, but rather to use *selective disabling*. Assume that the number of threads in the group being scheduled is smaller

than the number of processors under the controller. Then we need to disable only controllers situated along the paths from the scheduling controller to the processors used for threads of the current group; all other controllers in the subtree remain enabled. These active controllers can schedule other, smaller groups of threads. (As a special case, any processor left inactive can run any thread mapped to it.)

For example, if the top controller in a three-level tree schedules a group of five threads on processors 1-5 (see Figure 8), we should not disable the controller of processors 7 and 8; it can then use these two processors to schedule a two-thread group. Simulations show that selective disabling improves the run ratio of small groups, without any ill effect on large groups (see Figure 9). (We define the run ratio as the quotient of the execution time divided by the response time, that is, the fraction of the time the threads actually run.)

To evaluate the performance of a scheduling algorithm, we need a suitable metric. The metric must, of course, correspond to the original goals that guided the algorithm's design. Theoretical scheduling algorithms are usually designed to maximize throughput or minimize response time. To do so, they must have knowledge about the expected runtime of the various jobs. The distributed hierarchical control algorithm, on the other hand, does not assume any such knowledge. In addition, its design goal is the support of gang scheduling. Therefore, the metric for its evaluation should be the gang-scheduling efficiency achieved, that is, the percentage of

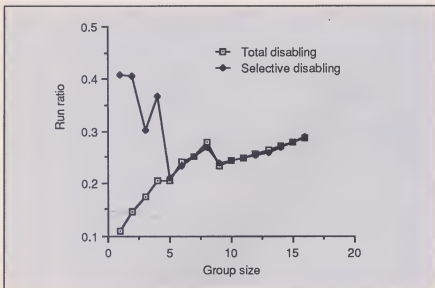


Figure 9. With selective disabling, previously wasted processing power runs small groups. The run ratio measures the fraction of the time that a group is actually executing. Large groups are unaffected.

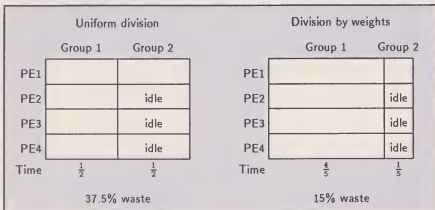


Figure 10. Example of scheduling groups with four and one threads. Waste is reduced if the time slices are determined by group weights, because the one-thread group that causes more waste is scheduled for less time. (PE: processing element.)

processors effectively used to gang schedule user tasks.

The only parameter in the algorithm we can modify to optimize this metric is the division of time between a controller and its subordinates. Interactive uniprocessor systems typically give the same time quantum to all ready jobs, with round-robin scheduling. (CPU-bound jobs may get longer quanta — and lower priorities — as they continue to execute, but a-priori all jobs get the same service.) Extending this to the distributed hierarchical control scheme is not trivial, because the number of groups of threads in distinct branches of the controller tree might differ. Therefore,

controllers must keep track of the maximal number of groups in any of the branches under them; they do this with the *groups* variable in Figure 6. In short, executing each group for a period proportional to $1/\text{groups}$ achieves uniform time slices.

Because of the gang-scheduling requirement, scheduling with such uniform time slices can be very inefficient. Consider an example in which the workload on a subtree with four processors consists of only two groups, one with four threads and the other with one thread. Using uniform time slices wastes 37.5 percent of the processing power, because when the single-thread group runs, three processors are left idle

(see Figure 10). Note, however, that the four-thread group does not waste resources. This points out a method for improving efficiency, namely, letting large groups run more than small groups.

Formally, the alternative to uniform scheduling is scheduling by *weight*. We define a group's weight as the number of its threads and a controller's weight as the total number of threads in all groups mapped directly to it. We can compute weight using the variable *threads* of Figure 6, as

$$\text{weight} = \text{threads} - (\text{threads}[\text{left-child}] + \text{threads}[\text{right-child}])$$

To divide the time according to weights, a controller computes the ratio of its weight to the total number of threads in its subtree, that is, $\text{weight}/\text{threads}$. This ratio gives the proportion of time allocated to its own groups. Simulation results indicate that using this scheme substantially reduces wasted processing power (see Figure 11).

An obvious drawback of dividing time by weights is that it impairs fairness. In the example given above, the single-thread group gets to run for only a quarter of the time that the four-thread group runs. However, this effect only exists when the workload is skewed, that is, when the various groups cannot fit together so as to use all of the processors. Had there been four single-thread groups in our example, their combined weight would have equaled that of the four-thread group, resulting in equal run ratios. Moreover, the use of selective disabling tends to favor small groups by giving them additional runtime. This more than compensates for the reduced time allocated in many cases, as shown in Figure 9.

Availability and fault tolerance

The trend in parallel operating system design has been from master-slave configurations towards symmetric configurations. At first, it was important for the system simply to work, even if it meant that a single failure (of the master) would bring down the entire system. After the art of getting systems to work was mastered, attention turned to making them more fault tolerant by, for example, using symmetrical systems. Replacing the single master with a tree of controllers achieves the same effect. As we show below, no controller is

critical for system operation. At most, a controller failure degrades performance.

The algorithms described in the previous sections are designed for a full tree of controllers supervising a set of processors whose number is a power of two. A real system, however, would have to contend with less ideal circumstances. Specifically, two situations we might expect are

(1) The number of processors is not a power of two. In large systems, the differences between successive powers of two are large, and it is unreasonable to disallow systems with intermediate sizes.

(2) There are not enough processors to satisfy a request. For example, the user might want to spawn more threads than there are processors.

The first problem has an algorithmic solution. In the original algorithms, controllers implicitly deduce the number of processors under their control by knowing their level in the tree. If the tree is not full, they must keep track of the number of processors explicitly. The same type of information filtration used to track loads in Figure 6 can be applied here. This knowledge is then taken into account in the mapping algorithms.

The solution to the second problem is to multiplex the processors. A too-large request propagates up the tree until it reaches the root. Since the root controller cannot pass the request further up, it maps the request itself. It splits the request into two requests and forces the subordinate controllers to care for them despite the fact that they are too large. The subordinate controllers continue this procedure until they reach the underlying processors. Because there are not enough processors, some of them will receive orders to care for more than one thread. When the order to gang schedule these threads later arrives, the processor must switch between them. Note that this is less expensive than conventional context switches, because threads belonging to the same group share all of their context except the processor state.

Even in a system with a complete tree and proper constraints on program requests, these two algorithmic complications can be useful—indeed, sufficient—in keeping the system operational in the face of fail-stop faults in any of the devices. We do not propose building a fail-safe system that guarantees correct termination of any job submitted despite runtime hardware failures; we make do with ensuring system availability at all times.

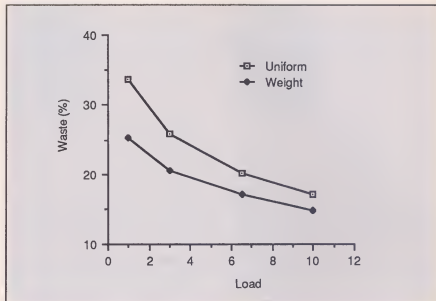


Figure 11. Simulation results show a reduction in wasted processing power when the scheduling times are determined by the weights of the groups of threads.

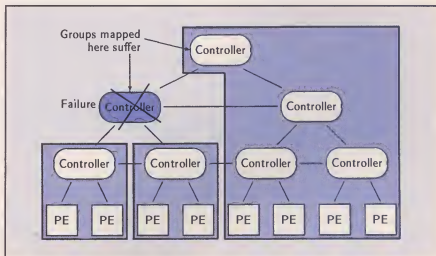


Figure 12. When a controller fails, it can be disconnected, leaving three operational, nearly independent systems. (PE: processing element.)

Thus, some jobs might crash if a device fails, but other jobs might not be affected. In addition, the system as a whole stays operational, so the crashed jobs can be submitted again without delay.

We achieve this type of fault tolerance as follows: When a controller fails, the tree is disconnected at the failed controller. Naturally, this requires special hardware support. This leaves us with three nearly independent systems: two small systems rooted at the direct subordinates of the failed controller, and the original system minus the subtree rooted at the failed controller (see Figure 12). Each of these sys-

tems can continue to function correctly, albeit with possible degradation of performance. The two small systems do so by multiplexing the few processors they have; the large system simply notes it is not a full tree. The three are not completely independent, because load balancing can still occur at the low levels.

Care is required, however, to handle failure on-line. To see that this simple scheme indeed works, note two additional points. First, a controller can fail while its subordinates are disabled, thus leaving their subtrees dormant forever. Since the subordinates mediate all interactions be-

tween a controller and its underlying processors, if the controller does not relinquish its hold on the processors for a long enough time (for example, a number of scheduling time quanta), the subordinates can revolt and regain control on behalf of the smaller groups of threads mapped on their subtrees. To do so, they simply assume the role of a root controller and start new scheduling waves in their subtrees. Second, when the failed controller is repaired, renewing the connections to it will restore the original full system (again, with the aid of special hardware support). The subordinates will find that they are not roots any more and can therefore send large requests upward. The superiors of the repaired controller will find a new branch and update their count of the available processors in the system.

An interesting question is what happens to those programs running when the fault occurs. The only groups of threads affected by a controller failure are those mapped to it or to one of its superiors, because such groups need the services of the failed controller to coordinate their gang scheduling. However, the absence of the controller does not imply that these groups cannot continue to compute. At most, they will suffer a degradation in performance because of the lack of gang scheduling. The only change required in the scheduling algorithm is that the failed controller's subordinates must allocate some time to the orphaned groups.

A processor failure is just a special case of a controller failure. Its main effect is to decrease the system size. The controllers need only note the new size. The main difference between a processor failure and a controller failure is that when a processor fails, the threads mapped to it cannot run any more. Thus, the groups to which these threads belong must be killed, and the user must submit the whole program again.

Other uses of distributed hierarchical control

We have proposed distributed hierarchical control as an operating system structure for controlling parallel processors. It is natural to ask whether we can use this control structure for other purposes. In this section we offer clues to its possible use in memory management and I/O.

Two of the most successful ideas in memory management for general-purpose,

multiuser computer systems are paging and virtual memory. Paging solves the fragmentation and overhead problems associated with contiguous allocation of memory and provides efficient division of memory among multiple jobs. Virtual memory allows for large address spaces that exceed the physically available memory and provides a level of protection between processes. It seems reasonable to expect these benefits in parallel systems as well.

The nature of a parallel system implies, however, that ideas that provided adequate performance in uniprocessors might not suffice. Specifically, we can expect more jobs to execute concurrently, each with a higher data-processing capability. Therefore, both the volume of the primary and secondary memories and the bandwidth of the communication between them must be much larger than is common today. A natural solution is to use numerous memory modules and disks in parallel, with an interconnection network between them. To ensure balancing, the address space should probably be interleaved across the devices.

Once we have many devices at hand, the question of coordinating their activities arises. Of course, coordination is not always necessary. For example, it seems senseless to create dependencies between distinct jobs that read or write different files from different disks. But we do need to coordinate the swapping of an address space that is interleaved across memory modules.

We thus face a dynamic system, where both independent activities and activities needing to coordinate arbitrary sequences of modules might appear. The concept of distributed hierarchical control is useful for dealing with such a system. Because the character of the memory-related activities depends strongly on the details of the paging and interleaving, this subject requires additional research.

for the allocation of memory and disk space.

Previous work in this direction falls short in a number of ways. The only system to support gang scheduling¹ with preemption was Medusa, which ran on Cm*.⁹ However, a central scheduler coordinated the system, so it was not scalable. The Mach operating system¹⁰ does not have truly scalable multiprocessor implementations either, although it has been ported successfully to small-scale machines such as the Encore Multimax. In addition, the scheduling servers implemented so far do not support interactive gang scheduling.

The distributed hierarchical control scheme introduced here constitutes a new concept in the field of parallel operating systems. (See the sidebar "Hierarchical control and gang scheduling" for a review of other projects involving some sort of hierarchical control.) This scheme provides a compromise between centralized master-slave-type systems, in which the master can become a bottleneck, and totally distributed systems, which lack global coordination. It is unique in combining the goals of being scalable, supporting interactive use, and fulfilling the paradigms of locality, simultaneous actions, and balancing. It meets these goals in a potentially simple and straightforward manner that also provides a degree of fault tolerance. When it is actually implemented, we can carry out a full cost/performance analysis. ■

Acknowledgments

This work was funded in part by a grant from the US-Israel Binational Science Foundation, 88-00045/1.

We would like to thank the guest editors of this special issue and the referees for their contribution to the improvement of this article.

References

1. K. Hwang and F.A. Briggs, *Computer Architecture and Parallel Processing*, McGraw-Hill, New York, N.Y., 1984.
2. J.L. Peterson and T.A. Norman, "Buddy

¹ Actually, it supported a variant of gang scheduling called co-scheduling. Co-scheduling attempts to schedule groups of related threads together, but makes do with scheduling only subgroups if not enough processors are available.

Systems," *Comm. ACM*, Vol. 20, No. 6, June 1977, pp. 421-431.

3. J.K. Ousterhout, "Scheduling Techniques for Concurrent Systems," *3rd Int'l Conf. Distributed Computing Systems*, CS Press, Los Alamitos, Calif., Oct. 1982, pp. 22-30.
4. A. Gottlieb, "Avoiding Serial Bottlenecks in Ultraparallel MIMD Computers," *Proc. Compcon Spring 84*, CS Press, Los Alamitos, Calif., 1984, pp. 354-359.
5. G.S. Almasi and A. Gottlieb, *Highly Parallel Computing*, Benjamin/Cummings, Menlo Park, Calif., 1989.
6. Y-T. Wang and R.J.T. Morris, "Load Sharing in Distributed Systems," *IEEE Trans. Computers*, Vol. C-34, No. 3, Mar. 1985, pp. 204-217.
7. V.M. Lo, "Heuristic Algorithms for Task Assignment in Distributed Systems," *IEEE Trans. Computers*, Vol. 37, No. 11, Nov. 1988, pp. 1384-1397.
8. F.C.H. Lin and R.M. Keller, "The Gradient Model Load Balancing Method," *IEEE Trans. Software Eng.*, Vol. SE-13, No. 1, Jan. 1987, pp. 32-38.
9. E.F. Gehringer, D.P. Siewiorek, and Z. Segall, *Parallel Processing: The Cm* Experience*, Digital Press, Bedford, Mass., 1987.
10. R.F. Rashid, "Threads of a New System," *Unix Review*, Vol. 4, No. 8, Aug. 1986, pp. 37-49.



Dror G. Feitelson is working towards a PhD degree at the Department of Computer Science of the Hebrew University of Jerusalem, Israel. His research interests include parallel operating systems, parallel programming, and optical computing. He authored *Optical Computing: A Survey for Computer Scientists* (MIT Press, 1988), which was selected by the Association of American Publishers as the best new Professional and Scholarly Publishing book in computer science for 1988.

Feitelson received a BS in mathematics, physics, and computer science in 1985 and an MS in computer science in 1987 (cum laude), both from the Hebrew University. He is a student member of the IEEE Computer Society, IEEE, ACM, and Commsac.



Larry Rudolph is currently on the faculty of the Department of Computer Science at the Hebrew University of Jerusalem, Israel. His research interests focus on understanding the power of parallel processing. These interests include the interplay between algorithms, programming languages, operating systems, and architectures for parallel processing and extend into the investigation of the power of parallelism in biological systems. In addition, he is interested in the use of optical computing as a means of overcoming the current deficiencies of parallel architectures.

In 1981 Rudolph received a PhD from the Courant Institute of New York University. His thesis introduced the idea of combining fetch-and-add requests within an Omega network of a shared-memory parallel processor. His work formed part of the basis for the NYU-Ultracomputer and the IBM RP3.

Readers may write to the authors at the Dept. of Computer Science, Hebrew University of Jerusalem, 91904 Jerusalem, Israel.



Have you heard about
our new periodical...

Transactions on Parallel and Distributed Systems?

For information on this, or any of
our ten optional periodicals,
circle number 200
on the reader service card.

IEEE COMPUTER SOCIETY

Membership/Circulation Dept.
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA 90720-1264
(714) 821-8380

IMS-91

CALL FOR PAPERS

The First International Workshop on Interoperability in Multidatabase Systems

will be held in Kyoto (the beautiful old capital city of Japan) along with IEEE Conf. on Data Engineering. Original papers are solicited on topics including, but not limited to:

Autonomy	Resolving semantic heterogeneity
Interdependence	Federated & Multi database tools
Data Definition	New paradigms, object orientation
Transactions	Queries & updates, spec. & proc.
	Interoperability of DBMS & KBMS

CHAIRPERSONS: Ahmed K. Elmagarmid, C.S. Dept., Purdue U., West Lafayette, IN 47907 USA, (317) 494-1998 & Yutaka Matsushita, Dept. Instrumentation, Kelo U., Hiyoshi, Yokohama, Japan, 81-44-63-1141 ext. 3564.

SUBMIT 7 copies of an extended abstract of at most 5 pages (troff/TEX accepted) to **Marek Rusinkiewicz**, Houston U., C.S. Dept., Houston, TX 77204-3475, USA (713) 749-4791. email: marek@cs.uh.edu or **Yahiko Kambayashi**, Kyushu U., CS-CE Dept., Hakozaki, Fukuoka 812, Japan, email: yahiko@csce.kyushu-u.ac.jp. fax: 81-92-641-1825.

PROGRAM COMMITTEE: Y. Breitbart (USA), B. Czejdo (USA), T. Furukawa (Japan), H. Garcia-Molina (USA), G. Gottlob (Germany), S. Heller (USA), S. Hikita (Japan), T. Kato (Japan), L. Kerschberg (USA), W. Litwin (France), **EUROPEAN COORDINATOR**, Y. Masunaga (Japan), E. Neuhold (Germany), P. Scheuermann (USA), G. Schlageter (Germany), A. Sheth (USA), H. Tirri (Finland), S. Uemura (Japan), R. Wang (USA), **LOCAL ARRANGEMENTS:** M. Yoshikawa, M. El-Sharkawi, M. Nagata, X. Zhong; **PUBLICATION:** A. Sheth; **FINANCE:** R. Martin, **REGISTRATION:** A. Datta; **PUBLICITY:** W. Perrizo.

Submission deadline:	September 15, 1990
Revised Extended Abstract due:	January 15, 1991
Acceptance notices:	December 10, 1990
Workshop dates:	April 8-9, 1991



The Seventh IEEE Conference on Artificial Intelligence Applications

Miami Fontainebleau Hotel, Miami Beach, Florida, February 24-28, 1991

PRELIMINARY CALL FOR PARTICIPATION



IEEE COMPUTER SOCIETY



THE INSTITUTE OF ELECTRICAL
AND ELECTRONICS ENGINEERS, INC.

General Chair

Se June Hong, IBM Research
Program Committee Chair
Tim Finin, Unisys

Tutorial Chair

Daniel O'Leary, University of
Southern California

Publicity Chair

Jeff Pepper, Carnegie Group Inc.

The conference is devoted to the application of artificial intelligence techniques to real-world problems. Two kinds of papers are appropriate: case studies of knowledge-based applications that solve significant problems and stimulate the development of useful techniques, and papers on AI techniques and principles that underlie knowledge-based systems, and in turn, enable more ambitious real-world applications. This conference provides a forum for such synergy between applications and AI techniques.

Papers describing significant unpublished results are solicited along three tracks:

- "Scientific/Engineering" Applications Track. Contributions stemming from the general area of industrial and scientific applications.
- "Business/Decision Support" Applications Track. Contributions stemming from the general area of decision support applications in business, government, law, etc.

Papers in these two application tracks must: (1) Justify the use of the AI technique, based on the problem definition and an analysis of the application's requirements; (2) Explain how AI technology was used to solve a significant problem; (3) Describe the status of the implementation; (4) Evaluate both the effectiveness of the implementation and the technique used.

Short papers describing systems in use (up to 1,000 words, extended abstract) will also be accepted for presentation in these application tracks.

- "Enabling Technology" Track. Contributions focusing on techniques and principles that facilitate the development of practical knowledge based systems that can be scaled to handle increasing problem complexity. Topics include all AI and programming techniques.

Papers should be limited to 5000 words. Papers significantly longer than this will not be reviewed. The first page of the paper must contain the following information (where applicable) in the order shown:

- Title.
- Author's name and affiliation (specify student status).
- Contact information (name, postal address, phone and email address).
- Abstract A 200 word abstract that includes a clear statement describing the paper's original contributions and what new lesson is imparted.
- AI topic: One or more terms describing the relevant AI areas, e.g., knowledge acquisition, explanation, diagnosis, etc.
- Domain area: One or more terms describing the problem domain area, e.g., mechanical design, factory scheduling, education, medicine, etc. Do NOT specify the track.
- Language/Tool: Underlying programming languages, systems and tools used.
- Status: Development and deployment status, as appropriate.
- Effort: Person-years of effort put into developing the particular aspect of the project being described.
- Impact: A 20 word description of estimated or measured (specify) benefit of the application developed.

Each paper accepted for publication will be allotted seven pages in the conference proceedings. The best papers accepted in the two applications tracks will be considered for a special issue of IEEE EXPERT to appear late in 1991. Application has been made to reserve a special issue of IEEE Transactions on Knowledge and Data Engineering (TKDE) for publication of the best papers in the enabling technologies track. IBM will sponsor an award of \$1,500 for the best student paper at the conference.

In addition to papers, we will be accepting the following types of submissions:

- Proposals for tutorial presentations. Proposals for the three hour tutorials of both an introductory and advanced nature are requested. Topics should relate to the management and technical development of useful artificial intelligence applications. Tutorials which analyze classes of applications in depth or examine techniques appropriate for a particular class of applications are of particular interest.

Each tutorial proposal should include the following:

- Detailed topic outline and extended abstract (about 3 pages).
- Intended audience and assumed background knowledge.
- Half-page synopsis of focus, topics, and benefits to audience.
- Full professional vita (including lecture/tutorial experience) and a one-paragraph summary.
- Proposals for panel discussions, 15-minute demonstrations (live or video), and 30-minute vendor presentations.

IMPORTANT DATES

- August 31, 1990: Six copies of papers, and four copies of all the proposals are due. Submissions not received by that date will be returned unopened. Electronically transmitted materials will **NOT** be accepted.
- October 23, 1990: Author notifications mailed.
- December 7, 1990: Accepted papers due to IEEE. Accepted tutorial notes due to Tutorial Chair.
- February 24-25, 1991: Conference tutorial program.
- February 26-28, 1991: Conference technical program.

Submit Papers and Other Proposals to:

Tim Finin
Center for Advanced Information Technology
Unisys
70E Swedesford Road
P.O. Box 517
Paoli, PA 19301
Phone: 215-648-2840
CSNET: finin@prc.unisys.com
Fax: 215-648-2288

Submit Tutorial Proposals to:

Daniel O'Leary
Graduate School of Business
University of Southern California
Los Angeles, CA 90089-1421
Phone: 213-743-4092
Fax: 213-747-2815

For registration and additional conference information, contact:

CAIA-91
IEEE Computer Society
1730 Massachusetts Avenue, NW
Washington, DC 20036-1903
Phone: 202-371-1013

STANDARDS

Editor: Fletcher J. Buckley, 103 Wexford Dr., Cherry Hill, NJ 08003, phone (609) 866-6350, fax (609) 866-7753, Compmail+ f.buckley

How not to write commercial standards

H. Ronald Berlack, Lockheed Sanders

In the fall of 1988, Robert Costello, US Under Secretary of Defense for Acquisition, got a mixed reception to his announcement that the Department of Defense was committed to procuring some 75 percent of its components from off-the-shelf commercial sources and, in doing so, would cease developing and writing specifications and standards relating to those components.

To many in the defense and commercial industries, this was good news. To the former, it meant relief from documents written in "governmentese" containing many restrictive idiosyncrasies that people in the industry felt were unnecessary. To the latter, it meant a chance to compete in an open market with less stringent controls, more flexibility, and the ability to push automated technologies to their fullest extent.

On the other hand, critics of Costello's decision warned that the DoD could not expect to receive the quality and reliability that the DoD standards had presumably enforced, even though many commercial equivalents survived and were operating in DoD-prescribed environments. (Automobile ignitions working without failure in the extremely frigid temperatures of northern Alaska provide a good example.) Despite this, the commercial standards writing bodies, also known as voluntary standards writing groups, warned the DoD that many of the voluntary standards were indeed not as detailed nor as accurate or assertive in quality and reliability test requirements as the DoD's were. Furthermore, each group had its own format and content requirements, and thus were inconsistent with the current DoD standards.

However, this did not deter the Defense Data Management Office under Peter Yurcisin from listing some 500 component standards to eliminate and for which the commercial industry would become responsible. Granted, most of these pertained to nuts-and-bolts type parts, sometimes called "jelly bean" parts or open stock items. But it did put the commercial world—in the US, in particular—on notice. The industry was being told to stress quality and reliability or risk losing competitive advantages to

overseas industries that could easily market parts through US distributors, even directly to the General Services Administration, without compliance with US standards.

The "what" versus the "how." During the past year, the commercial world has been working closer to the ideals of technical quality management (TQM), statistical process control (SPC), and the use of just-in-time (JIT) inventory techniques. But, so has the defense industry.

Interestingly, while defense industry associations have pressed for discarding counterproductive standards, adopting specifications with less restrictions, and

produce in accordance with "how to" requirements rather than being left to his or her own methods of production.

The bottom line must be that it's the *what* that's important, not the *how*. This is particularly critical when we discuss software-development requirements. The same issues discussed above also pertain to software development. In fact, there is even more urgency in considering the *what* versus the *how* due to the volatility of software development and software languages, much less the platforms the software operates on and the environment that supports it.

In the mid 1970s, the US Navy was very concerned that it was acquiring software systems by various means with the standards then on hand and, in turn, receiving many different varieties of systems, none of which had any commonality whatsoever. It was one of those situations where a common-use routine could be ordered from several software developers to ostensibly the same specification and was received in several different versions and in varying grades of quality.

To prevent this, the Navy developed and issued a standard for defense systems software development that was intended to insure that all software was ordered and developed in accordance with the standard and then was tested, inspected, and received in a manner that met the requirements of the standard. The first draft of the document drew more than 4,000 comments from both industry and government.

The greatest complaint was that the standard told the user how to develop, test, and inspect the product as it evolved. So many diverse software environments existed among the many developers, much less users, that the standard could not be uniformly applied unless all those impacted trashed their current environments and procured those that conformed to the naval standard. And, were this done, would the Navy or governmental agency pay for it?

The answer, of course, was "No," and a great deal of work went into refining the draft standard to make it as uniformly applicable as possible. A subsequent revision went several steps further to placate

The bottom line must be that it's the *what* that's important, not the *how*.

This is particularly critical when we discuss software-development requirements.

eliminating the "how to do it" syndrome, commercial industry associations appear headed in the other direction. In addition, recent publications seem to contain far more restrictive provisions and a good deal of the "how to do it" as well.

An example of this is a standard published by a well-respected commercial industry association. This document stated that "the management of an organization shall develop and document its management control policy... Management shall ensure that this policy is understood, implemented, and maintained."

This should greatly concern the DoD as well as commercial industries and their customers. Several studies conducted within DoD environs have demonstrated increasing costs when one is forced to

the how-to issues, but even that didn't eliminate all of the how-to statements.

While the software industry was arguing for relief from the how-to, the US Army, distressed about the quality of the software it acquired, developed a specification on quality requirements. While not as full of how-to instructions as the Navy standard, the Army's specification still contained some tight requirements the software developers were not used to. In spite of arguments that the specification was a cost driver, the Army proceeded to publish, issue, and impose the document on its contractors.

These two instances were repeated with the advent of tri-service initiatives for a more compatible set of standards for software and software quality. The argument for tailored provisions and less complicated documents that spell out only the what, complemented by implementation guides that describe the how-to, is still going on. The adage that you can please some of the people all of the time but cannot please all of the people all of the time will continue between developer and government for years to come.

Question emerges. An interesting question has arisen: If software developers are so adamant about wanting to be left alone to follow their own methods and implement what they believe is the best and latest technology, why are we suddenly seeing commercial specifications and standards with the same type of how-to instructions that prevailed in the government documents?

I suspect one motive is the writer's feeling of uncertainty that the reader will understand what is being said or what is being specified. I have seen a number of instances in source as well as specification procurement drawings where writers felt they had to instruct the vendor in how to test the items being ordered. It got to the point where they had to literally specify the brand of equipment to be sure the product would pass incoming inspection and not have to be returned and consequently cause schedule delays.

This, of course, is a normal, well-intentioned concern, but does nothing to alleviate the cost of imposing how-to instructions on vendors, much less software developers.

On the other hand, several very good standards from several different writing groups do very well in specifying the what and follow up with the how in separate appendices or guides. These include the IEEE standards for designing, developing, building, testing, and delivering software, as well as those related to electrical and electronic component activities. The Electronic Industries Association's ANSI/RS Standards (for example,

for RS-232) and EIA standards for electronic components are also well reviewed by the membership, as are those for the Society of Automotive Engineers and a number of other groups.

In each case, the various associations and writing groups have standards and guides for format and content that are very specific in how to develop and write a standard. But I'm concerned with those people writing standards and guides who don't author them well enough to insure adherence and prevent pitfalls.

Avoiding how-to. The following discussion tells you what to do to avoid that ghastly "how to" phrase. We will look at a common product specification/standard used in developing a product. While this may seem more hardware than software oriented, it is equally applicable to the

The general and detailed requirements sections form the crux of the document. If you have set the tone correctly ... these two sections can flow quite well ... using such terms as "shall," "will," and "may."

software environment. This will be demonstrated by comparisons to IEEE standards for software.

The format for a typical standard usually entails six parts:

- 1.0 Scope, purpose, definitions, and applicable documents.
- 2.0 General requirements.
- 3.0 Detailed requirements.
- 4.0 Test requirements, quality evaluation.
- 5.0 Packaging and shipping criteria.
- 6.0 Notes, appendix, index, etc.

When writing software standards, author(s) may not always follow this six-part progression of data, but they'll usually discuss the subject matter in roughly the same order as shown.

The content and focus of section 1.0 may look benign, but can cause a lot of consternation and apathy if not treated carefully. We sometimes forget that this is a reader/user's first look and, as we were all taught, first impressions are important.

You'll encounter the biggest pitfall in the first sentence of scope (1.1) if you write "This standard is mandatory for the development of compiler routines." If you feel guilty about writing it this way, you might change it to "This standard must be used for developing all compiler routines." Reader responses to such statements may vary. But, in essence, readers may conclude there are better ways to develop compiler routines and choose another way. Should this happen, the writer will have been defeated in developing a standard approach.

Normally, the first sentence of scope will read "This standard provides the criteria for the development of compiler routines." The simplicity of the sentence should convey that readers are about to learn the standard way to develop these routines.

The purpose segment (1.2) can also be problematic. If the document's title contains the word "standard," the purpose section should discuss criteria, norms, or commonality arrived at by consensus and must avoid such terms as "mandatory," "must," "will not," and "cannot." The word "specification" suggests the section will define, require, identify, or detail. Thus, in section 1.2 we would expect to read something like "The purpose of this standard is to establish a common way to develop compiler routines for the Ada language."

The definitions section (1.3) is also vulnerable if you have restricted its scope to a very specific set rather than providing definitions most commonly appearing in standard dictionaries or glossaries. Granted, it's possible to use definitions that only apply to this document, but their fate is usually decided during the review and approval phases. You must avoid writing a preamble that states that these are the only acceptable definitions.

The applicable documents section (1.4) can also cause problems if its preamble states that "the following documents must be adhered to" or that "the development of compiler routines must be performed in accordance with the following documents." Even government standards and specifications limit their preambles to "The following documents of record at the time of contract award are applicable..." IEEE standards are most careful to state that "The following documents may be followed in the development of..." I emphasize the word "may" because IEEE standards are published for voluntary and not mandatory use.

Heart of the document. The general and detailed requirements sections (2.0 and 3.0) form the crux of the document. If you have set the tone correctly in section

I.O, these two sections can flow quite well with the use of such terms as "shall" and "will" and, where best suited, "may." Again, the tone should be handled in such a way that they don't force someone into doing something costly, unwarranted, or unnecessary.

For instance, a recent standard on integrated circuit manufacturing stated, "A technical review board shall be formed and is responsible for the development of [a] TQM plan... The manufacturer's technical review board shall consist of, as a minimum, representatives of..." A better way to put it would be "The manufacturer shall describe in a plan how TQM will be monitored and maintained." Some would argue that even that version sounds like "how to" and would prefer simply "The manufacturer shall perform TQM..." In either case, consensus of the reviewers and final approval authority should govern how the requirements should be stated. If you feel more detailed instruction is needed, such data can be placed in the appendix in section 6.0 or in a separate volume or guide.

Careful writing is important to avoid unnecessary requirements that can be costly to implement or meet and have no intrinsic value for successful adherence to the standard. This area has received a great deal of attention over the past several years because studies have shown that counterproductive requirements can lead to unnecessary and excessive costs. One of the best arguments is: Why is only one specification document required for constructing a 747 jet airliner, but 14,000 such documents are needed to build the US President's new Air Force One, which is also a 747?

The next section, 4.0, can also be loaded with how-to requirements. While this section is far more of a problem when it comes to hardware items, it is also a problem with respect to software, especially when the authors are unsure how a software element should be tested and evaluated.

One of the more onerous passages appearing in a standard was a requirement in the Navy primary document for software development that stated, "Regression testing shall be conducted on all changes to approved software elements." This amounted to pure overkill that, however desirable, was costly. Although later rescinded in revision A, it could have been handled through other, cheaper means.

It is, therefore, important to describe the test criteria the software will have to meet while being careful to avoid statements on how the developer must perform the test or the evaluation. How the developer will test to ensure the software is portable to three different databases is the developer's problem, and he or she

will have to resolve it through demonstration or other means.

The section on packaging and shipping, 5.0, shouldn't be problematic. However, it should be examined to ensure requirements for installation and checkout at the customer's site are not placed here but in the requirements sections.

Section 6.0, which shouldn't affect the previous sections, is where you can describe optional how-to procedures to satisfy requirements and include examples.

The four appendices of *ANSI/IEEE Standard 1042, A Guide for Software Configuration Management* serve as a good example. These sections contain fully written software configuration management plans the developer can follow in meeting the requirements of *ANSI/IEEE Standard 828*. Another good example is *ANSI/IEEE Standard 830, A Standard for Software Requirements Specifications*, which provides several options for content format the developer can elect in meeting basic requirements spelled out in the body of the document.

The primary lesson. The principal lesson to be learned from this discussion is that, to avoid the specter of how-to, you must keep the standard or specification simple, no matter how complex the task.

Describing application environment profiles as a significant tool for simplifying and coordinating standards efforts, Jim Isaak stated¹ that "An AEP needs to be complete with respect to its environment. That is, it should reference all of the standards required and speak to all of the options. Where an essential function does not exist, this needs to be identified and a course of action needs to be recommended."

In a series of articles in *IEEE Software*,^{2,3} Bob Poston provided excellent examples using Standard 830 to keep standards and specifications simple, to the point, unambiguous, and readable.

If you describe more "what" in simple terms and avoid long toms of specifications inundated with how-to descriptions, you'll create more economical and trustworthy software.

References

1. J. Isaak, "Applications Environment Profiles," *Computer*, Feb. 1990, pp. 69-70.
2. R.M. Poston, "Preventing Software Requirements Specification Errors with IEEE 830," *IEEE Software*, Jan. 1985, pp. 83-88.
3. R.M. Poston, "Workday Software Requirements Specifications," *IEEE Software*, Sept. 1985, pp. 63-65.

The Swiss Federal Institute of Technology (ETH Zurich) has an opening for a

Professor of Computer Science

The fields of specialization of particular interest to the Department of Computer Science are:

- Distributed Systems
- Parallel Computing
- High Performance Data Bases, Parallelism
- Algorithms for Robotics
- Computer Graphics

Duties of the new professor will also include teaching at the undergraduate and the graduate level.

Candidates are invited to submit their application together with a curriculum vitae, a list of publications, names of referees, and a description of their research activities and plans before July 31, 1990, to the President of ETH Zurich, Prof. Hans Bülmann, ETH Zentrum, CH-8092 Zurich, Switzerland.

For further information please contact Prof. H.P. Frei, Chairman of the Department of Computer Science, ETH Zentrum, 8092 Zurich, Switzerland, fax number: +41-262-3973.

**Late Magazines?
No Magazines?
Membership
Status Problems?
No Answers
To Your
Complaints?**

**Let your
Computer
Society
Ombudsman
cut
through
the red
tape
for you.**



THE COMPUTER SOCIETY
Ombudsman
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
COMPMAIL + CS HELP

Board of Governors adopts statement on proposed IEEE volunteer reorganization

I. Mark Haas, Chair, Planning/Self Assessment Committee, Conferences and Tutorials Board

During the March 2 meeting of the IEEE Computer Society Board of Governors in San Francisco, the board adopted a statement as drafted and recommended by the society's Executive Committee regarding the volunteer reorganization of the IEEE. The statement addresses proposed IEEE "national entities" and ways to enhance the international focus of institute and society programs, the organizational status of the IEEE-USAB organization, standards, and educational accreditation.

The board's position statement views the IEEE proposals in these areas as unnecessarily drastic responses to minor or nonexistent problems. While citing the effectiveness of current approaches, the position statement explains why the board sees elements of the IEEE proposals as inappropriate, and in some cases suggests less drastic alternatives.

The position statement was the subject of Computer Society President Helen Wood's column in the April issue of *Computer* (pp. 4-6), and the complete statement appears there.

The board also addressed a number of other issues during this meeting.

Finances. Treasurer Joseph Boykin announced that the Computer Society's 1989 deficit will be larger than anticipated last November when the 1990 budget was adopted. The revised deficit figure for 1989 will be approximately \$280,000. Boykin noted that the society's cash reserves have dropped below the guidelines set by the Board of Governors.

Appointments. Computer Society President Wood announced the election by the IEEE of J. Thomas Cain as IEEE

Division VIII (Computer Society) director-delegate. Cain is replacing Roy Russo, who resigned as Division VIII director-delegate.

The board also approved the appointment of Bill Carroll, Tadao Ichikawa, Ray Miller, Harold Stone, and Don Thomas to the IEEE Computer Society Audit Committee, as recommended by Ken Anderson for the Nominations Committee, which he chairs. The Audit Committee elected Harold Stone to serve as its chair.

Amendments. The bylaw amendment to restructure the Finance Committee to improve efficiency of the budget development process (see *Computer*, January 1990, p. 81) received a second approval by the Board of Governors.

Publications. The Board of Governors approved a second pilot issue of *Computing Futures* for publication in 1990. The first issue, published in late 1989, was put together by students and distributed to Computer Society student members in northern California. The 1990 issue will have a student editorial board of wider geographic scope and will be distributed internationally to every student member of the Computer Society. In addition, a task force has been formed to develop a proposal to turn *Computing Futures* into an ongoing publication.

Sallie Sheppard, vice president for publications, reported that a survey is in progress to determine member interest in a proposed advanced abstracts periodical.

Press. The board approved spending \$50,000 for a joint project between the Press Activities Board and the Publica-

tions Board to investigate alternative delivery mechanisms and new media for society publications. Vice President Jim Aylor noted three areas targeted for initial investigation: advanced abstract services, delivery of society magazines on floppy disks or CD-ROMS, and single-copy article delivery after an on-line database search.

Membership. Vice President Barry Johnson announced that Computer Society membership reached an all-time high of 106,674 in 1989. While both regular and affiliate membership are up, student membership has declined slightly.

Conferences and tutorials. Vice President Laurel Kaleda informed the board that Conferences and Tutorials has noted a recent industry downturn that will probably impact conference attendance and that the C&T Board is monitoring the situation closely.

Awards. The board approved a proposal for a new award, the annual Young Computer Engineer/Scientist Award, to recognize outstanding contributions to the profession. The proposal will be forwarded to IEEE for final approval.

Anniversary activities. Next year will mark the 40th anniversary of the IEEE Computer Society. Stephen Yau, History Committee chair, announced several special activities for 1991 to celebrate the milestone. These include special 40th anniversary receptions at several conferences, a special 40th anniversary issue of *Computer*, and a series of 40th anniversary prizes to be awarded for service to the society.

Award winners honored at Compcon

Prominent among the activities at Compcon Spring 90, February 26-March 2 in San Francisco, was the IEEE Computer Society's presentation of awards honoring outstanding achievements within the computer industry as well as significant contributions to the society. The ceremony was held in conjunction with an awards luncheon.

The 1990 IEEE Emanuel R. Piore Award was presented to Allen Newell for "seminal contributions to artificial intelligence." The award includes a \$2,000 honorarium and a \$2,500 international travel grant. Newell is the U.A. and Heler Whittaker University Professor of Computer Science at Carnegie Mellon University. He began his research on artificial intelligence in 1954.

Victor Basili, Sakti P. Ghosh, David Patterson, Bruce Shriver, and Shanker Singh received IEEE fellow certificates. (In the formal announcement of Computer Society members who became IEEE fellows — *Computer*, February 1990, pp. 72-73 — Singh was inadvertently omitted.)

The Computer Society's Computer Pioneer Award this year honored seven people "whose efforts resulted in the creation and continued vitality of the electronic computer industry." The annual award goes to outstanding individuals whose main contribution to the concepts and development of the computer field was made at least 15 years earlier.

Recipients for 1989 included John Cocke for instruction pipelining and RISC concepts, Ralph L. Palmer for the IBM 604 electronic calculator, and James A. Weidenhammer for high-speed I/O mechanisms.

A special Computer Pioneer Award was made jointly honoring Mina S. Rees, F. Joachim Weyl, Marshall C. Yovits, and Gordon D. Goldstein of the Office of Naval Research for "vigorous support of computer research and development beginning in 1946." Weyl's award was accepted by his widow. Goldstein received his award in April 1989, shortly before his death (see *Computer*, June 1989, p. 98).

Edward B. Eichelberger and Thomas W. Williams of IBM received the W. Wallace McDowell Award for "developing the level-sensitive scan technique of testing solid-state logic circuits and for leading, defining, and promoting design for testability concepts." The award was established through a grant by IBM and is presented annually by the Computer Society to individuals whose professional work has been outstanding in concepts, technology, programming, education, or

management in the computer field.

The Computer Entrepreneur Award for 1989 was presented to Gene Amdahl in recognition of his entrepreneurial efforts in the development of a strong and competitive mainframe computer industry. Amdahl spent the early part of his career with IBM before leaving in 1970 to form and operate his own company. More recently, he founded Andor International, a computer systems company.

The Richard E. Merwin Award for Distinguished Service was presented to Stanley Winkler "in recognition of outstanding contributions to the computer profession."

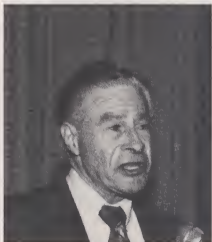
Winkler, whose involvement with computers began in 1942, was with IBM from 1959 until retirement in 1983 and served the company in various management and technical positions. He is currently a guest research worker at the Na-



Allen Newell received the IEEE Emanuel R. Piore Award for his contributions to artificial intelligence.



James A. Weidenhammer received the Computer Society's Computer Pioneer Award for his work with high-speed I/O mechanisms. Computer Pioneer Awards also went to John Cocke and Ralph L. Palmer (not shown).



A special Computer Pioneer Award recognized the efforts of Marshall C. Yovits and Mina S. Rees of the Office of Naval Research, who were given the award jointly with F. Joachim Weyl and Gordon D. Goldstein, both deceased.

tional Institute for Standards and Technology, where he pursues his interests in very large concurrent systems and computational science.

The Merwin Award, first presented in 1981, is given annually and consists of a certificate and a \$1,000 honorarium.

Service to the society. An Outstanding Contribution Award went to *IEEE Expert* Managing Editor Henry Ayling for serving as managing editor of the inaugural issue of *IEEE Computing Futures*. Ayling also received a Meritorious Service Award for outstanding contributions to the success of *IEEE Expert* and activities

related to desktop publishing.

Also receiving a Meritorious Service Award were David Choy, for technical and administrative leadership while serving as chair of the Technical Committee on Office Automation, and Ralph Preiss, for outstanding leadership as chair of the Awards Committee from 1983 to 1987. In addition, Preiss received an Outstanding Contribution Award for contributions to the Committee on Public Policy's activities, for serving as the 1985-87 chair of the Awards Committee, and as founding editor of the COPP newsletter.

Certificates of Appreciation were presented to the following:

- Ronald Braun for "significant technical and international standards liaison contributions as secretary of the Draft Standard on Methodology for Software Quality Metrics Methodology."

- Fred Buelow for "outstanding efforts as chair of the W. Wallace McDowell Subcommittee, thus helping to ensure the continuation of a high-quality and significant awards program."

- Paul Davis for "continuous and diligent service as a member and secretary of the Committee on Public Policy."

- William Eventoff for "contributions leading to the successful revision of ANSI/IEEE Standard 730.1 - 1989, IEEE Standard for Software Quality Assurance Plans."

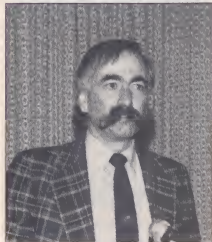
- Harry Huskey for "outstanding efforts as chair of the Computer Pioneer Subcommittee, thus helping to ensure the continuation of a high-quality and significant awards program."

- Willis King for "significant service to the profession in the Computer Society Accreditation Board's accreditation process."

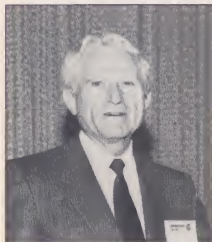
- Celia Modell for "contributions leading to the successful revision of ANSI/IEEE Standard 730.1 - 1989, IEEE Standard for Software Quality Assurance Plans."

- Arthur Pohm for "outstanding efforts as chair of the Computer Entrepreneur Subcommittee, thus helping to ensure the continuation of a high-quality and significant awards program."

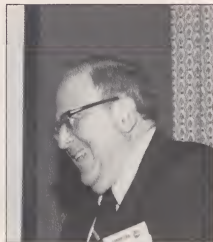
- James Slagle for "technical and administrative leadership while serving as chair of the Technical Committee on Computational Medicine."



The W. Wallace McDowell Award was presented to Thomas W. Williams (left) and Edward B. Eichelberger for their work with test techniques.



The Computer Entrepreneur Award went to Gene Amdahl for his business ventures in the mainframe computer industry, beginning in 1970 with the founding of Amdahl Corp.



Stanley Winkler received the Richard E. Merwin Award for "outstanding contributions to the computer profession."



Henry Ayling, managing editor of *IEEE Expert*, received an Outstanding Contribution Award for serving as managing editor of the inaugural issue of *IEEE Computing Futures*.

Staff awards. Additional awards were made to Computer Society staff members February 22 during a meeting in the West Coast publications office attended by Computer Society President Helen Wood and Executive Director T. Michael Elliott.

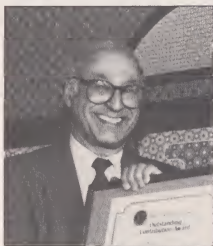
IEEE Design and Test of Computers Managing Editor Nancy Talbert received an Outstanding Contribution Award for leading the experiment to produce transactions via desktop publishing technology in the society's publications office.

Patricia Paulsen, assistant to the publisher, received a Meritorious Service Award for playing a key role in processing the Computer Society awards.

Marilyn Potes received a Certificate of Appreciation for providing dedicated and consistent support as managing editor of *Computer* magazine, 1987-89. She was recognized for "diligent and untiring assistance with the President and Executive Committee messages published in *Computer*."

In the society's East Coast office, Magdalene (Maggie) Johnson was pre-

sented with a Meritorious Service Award for dedicated and consistent high-quality service to the conferences and tutorials



Ralph Preiss received an Outstanding Contribution Award for chairing the Awards Committee from 1985 to 1987 and for his efforts on behalf of the Committee on Public Policy.

program. She was cited for diligence and courtesy in her work with volunteers in creating and analyzing meeting budgets.



Nancy Talbert, managing editor of *IEEE Design and Test of Computers*, received an Outstanding Contribution Award for leadership in a desktop publishing feasibility project.

Gordon Bell Prize winners announced at Compcon

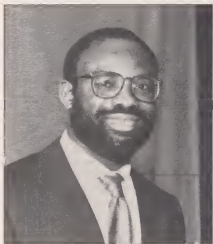
Winners of this year's Gordon Bell Prize, which is actually three cash prizes of \$1,000 in the categories of raw performance, price/performance, and compiler speed-up, were announced by *IEEE Software* Editor-in-Chief Ted Lewis during the IEEE Computer Society's awards ceremony at Compcon. There was no winner in the compiler speed-up category this year.

In the category of raw performance the winner was a team from Mobil Research Laboratories and Thinking Machines Corp. In describing the work of the winning team members on finite-difference seismic modeling, Alan Karp, one of the judges, said, "They used a reasonable algorithm on a real problem of reasonable size. The performance is truly outstanding." The application achieved a rate of 5.6 gigaflops.

Team members are Doug McCowen and Irshad Mufti of Mobil Research and Mark Bromley, Harold Hubschman, Alan Edelman, Bob Lordi, Jacek Myczkowski, and Alex Vasilevsky of Thinking Machines.

The price/performance prize went to Philip Emeagwali of the University of Michigan for his work on oil-reservoir simulation. Emeagwali's effort achieved 365 megaflops per million dollars.

"I've checked with several reservoir engineers who feel that his calculation is of real importance and very fast," said



Philip Emeagwali won the Gordon Bell Prize in the price/performance category for his work on oil-reservoir simulation. A team consisting of Doug McCowen, Irshad Mufti, Mark Bromley, Harold Hubschman, Alan Edelman, Bob Lordi, Jacek Myczkowski, and Alex Vasilevsky won in the raw-performance category.

Karp. "His explicit method not only generates lots of megaflops but solves problems faster than implicit methods. According to my contacts, no one has applied the pseudotime approach in reservoir modeling."

The prizes are sponsored by Gordon Bell, vice president of engineering at Ardent Computers in Sunnyvale, California. They are awarded in recognition of superior efforts in practical parallel processing rather than for academic research.

Two \$250 honorable mentions also were awarded. The first was to Sunil Arvindam, Vipin Kumar, and V. Nageshwar Rao of the University of Texas at Austin for their work on search problems. The second went to the team of Daniel Lopresti of Brown University and William Holmes of the Institute for Defense Analysis Supercomputing Research Center for their work on a reconfigurable linear logic array.

Winners were selected by a panel that included Karp of the IBM Palo Alto Scientific Center, Jack Dongarra of Oak Ridge National Laboratory, Ken Kennedy of Rice University, and David Kuck of the University of Illinois. *IEEE Software* Associate Editor-in-Chief Shreekanth Thakkar coordinated the judging.

Newly established TC provides a forum for practical AI

Sandra Hoffman, Vice Chair for Communications, Technical Committee on Expert Systems Applications

Chartered as a task force in March 1989 and elevated to a technical committee last November, the IEEE Computer Society's TC on Expert Systems Applications has developed rapidly in its first few months.

This rapid rise in status is due largely to the organizational skills of founder Dan Yurman, who chaired the group until October, and to the enthusiasm he generated. New Chair Harry Siegel presented the successful proposal for TC status to the Technical Activities Board.

Yurman's legacy includes a membership of 1,000; a quarterly newsletter; active committees on standards, industrial relations, and technology transfer; and sponsorship of two national conferences.

Objectives. Why did this group grow so quickly? To quote from the proposal to the TAB:

The objectives of the technical committee are to improve the abilities of organizations and individuals to work with expert systems technologies... Expert systems are the most mature and resilient products to emerge from the AI community, and they are being adopted by corporations and government departments to improve productivity. They

are doing this because the applications of expert systems to specific knowledge-intensive systems return high yields. Success stories for expert systems are more common now than two years ago. By current estimate there are 2,000 operational expert systems, and 80 percent of them are running on personal computers.

The greatest value to be derived from participation in the technical committee will come from regular discussions among participants. In some ways, this will resemble the informal interactions of a user group. In other ways it will complement many of the professional activities of the IEEE Computer Society.

Meetings and membership. TC members have varied backgrounds in private industry, military and civilian government agencies, and universities, and speakers at the meetings reflect this diversity. Meetings in Washington, DC, have drawn a mix of system developers with stories to tell and managers who want to learn how expert systems can help them and how to avoid common mistakes. Meetings in other cities will be encouraged so that forums can be provided for members (nearly one third) outside Washington. Randy Manner, (703) 841-6849, is vice chair for meetings.

An informal survey by the Standards Committee emphasizes the practical outlook of the TC membership. Members want guidelines to help them through each stage in the development of a system, including project selection, development methodologies, documentation, integration with other systems, and sharing of lessons learned. Vice Chair Roger Boan, (513) 429-4311, is setting up working groups on these and other topics.

Conferences. The Conference Committee is participating in AISIG 90, the Fifth AI Systems in Government Conference, May 7-11, in Washington, DC. The committee is also sponsoring an international conference on practical approaches to managing, developing, and institutionalizing expert systems, scheduled for September 10-12 — the IEEE Conference on Managing Expert System Programs and Projects. Also to be held in the Washington, DC, area, this conference will include topics such as cost-justifying expert systems, integrating expert systems into the workplace by training users and providing support and documentation, system development from the viewpoint of the expert on whose knowledge the system is based, selecting and training staff to build and maintain expert systems, and currently available PC and mainframe tools. Contact Vice Chair

Jerry Feinstein, (703) 821-0701, for more information.

Communications. The Communications Committee is concerned with sharing information among TC members and other professional organizations. Our quarterly newsletter, *The Inference Engine*, has published case studies of expert systems projects in the Internal Revenue Service and Navy Finance Center and is soliciting notable expert systems experiences (good and bad) for the "How Did It Work?" column. Rodger Knaus, (301) 530-0898, is newsletter editor and is also in charge of the new computer bulletin board. The number for this RBBS-based BBS is (301) 530-2890, available at 1200 and 2400 baud, with eight data bits, one parity bit, and null parity.

Other activities. The Industry Relations Committee has subcommittees focusing on private- and public-sector consumers and on vendors of PC-based and non-PC-based tools. Members will be writing product reviews for AI publications. In addition, a poster session, "Expert Systems in Industry," will be presented during one of the TC meetings. Vendors will exhibit development products and users will exhibit fielded systems. Joseph Schmuller, (703) 968-0900, is vice chair.

Linking university-based expert systems research — especially the results of fielded, successful systems — to potential users is another project of the TC. The Committee on Technology Transfer is developing relationships with organizations such as the American Association for the Advancement of Science, the Harvard Institute for International Development, the United Nations Development Program, the Food and Agriculture Organization (Rome), and the US Department of Agriculture. Steve Ruth, vice chair, (703) 323-2738, and Rizard Michalski have developed several modules for expert systems training for US as well as international students, faculty, and administrators.

The new TC has many enthusiastic people working to make AI technology helpful in day-to-day situations. We welcome anyone with like interests to join us. Our next Washington-area meeting is scheduled for June 6. Contact Harry Siegel, Jaycor, 1608 Spring Hill Rd., Vienna, VA 22182, phone (703) 847-4000, for more information.

TAB Coordinator Lori Rottenberg deserves a special word of thanks. Her skills and patience eased our growing pains.



Have you heard
about...

Transactions on Knowledge and Data Engineering?

For information on this,
or any of our ten optional periodicals,
circle number 200
on the reader service card.

IEEE COMPUTER SOCIETY

Membership/Circulation Dept.
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA 90720-1264
(714) 821-8380

PRODUCT REVIEWS

Editor: Richard Eckhouse, UMASS-Boston, Harbor Campus, Boston, MA 02125, Compmail+, r.eckhouse; Bitnet, eckhouse@umbusky; Compuserve, 70516,556

This issue deals with converting our thoughts into print. It represents the combined efforts of Daniel McAuliffe, Sorel Reisman, Bruce Shriver, Curtis Swanson, and myself in looking at the new breed of graphical word processors, laser printers, Postscript output, and printer utilities. We hope you will find the similarities and diversity as interesting as we did in evaluating the products. — Richard Eckhouse

Graphical word processors

The latest buzzword is GUI — graphical user interface. As applied to word processing, it means creating a document and seeing how it will look before sending it to the printer. Our reviewers examined the three most popular word processors and report here on their findings.

Ami Professional 1.1

When first reviewed in *Computer*, Ami from Samna was labeled a "dynamite product" in a review entitled "Word processing at its best" (see the June 1989 issue, pp. 101-102). That was a pretty powerful endorsement, one recently challenged by the introduction of Word for Windows (see the review below). What attracted us to Ami was its graphical approach to word processing, its intuitive pull-down menus, and the sheer fun of using it to produce documents with multiple typefaces, columns, and graphics.

Having already mastered Ami's features, we did not find the learning curve for the professional version very steep. We liked the promise of Ami Professional because of a number of features we really wanted that were not part of the original release. In particular, we liked

- the ability to view and edit two documents at once,
- a document description facility to supplement the limited eight-character DOS file names,
- an undelete function,
- mail-merge to insert text into a document,
- an expanded search and replace function,
- a choice of sidebar shortcut icons for frequently executed functions,
- footnotes and endnotes,
- a vertical ruler,
- an easy way to change lowercase words into uppercase,
- a macro facility,

- tables (a worksheet-oriented function for formatting tabular information, including a basic math capability),
- a built-in drawing program,
- a sort capability,
- the ability to export various text and word processing file formats,
- a thesaurus, and
- a glossary for "boilerplate" text.

An unexpected benefit was that the developers seemed to have tweaked out a bit more speed.

Speed, or lack thereof, is one of the main drawbacks of Windows-based word processors. Fortunately, both standard and professional versions of Ami have a nongraphics draft mode for rapid input of keystrokes. We also found that setting up Windows for optimal use of expanded or extended memory is crucial for acceptable performance.

Why Ami Professional? Ami

Professional's value lies in its ease of use combined with a number of high-end features. These features, in addition to those already mentioned, include

- bookmarks,
- table of contents and index generation;
- more control of headers and footers;
- document annotation;
- charting of numeric data;
- text changes to show strikethrough, double underline, and small caps;
- image processing to enhance scanned images;
- dynamic data links to Microsoft's Excel;
- line numbering; and
- a "go to" feature.

Other features, such as style sheets and widow-orphan control in both versions, directly target desktop publishing requirements. The professional version includes a powerful macro language and a

number of useful macros for batch printing, word counting, recall of recently edited documents, and maintenance of a merge file, just to name a few. These macros add flexibility to day-to-day operations and demonstrate how to expand what Ami Professional can do (like link to Excel using dynamic data exchange or change the pull-down menus).

The documentation has not changed. While sufficient to get you started, it is terse. Often, you can't find a term in the index to learn more about a topic that needs additional explanation. As a result, the system forces you to try features rather than read about them. In a sense, experimenting is fun, but it can be equally frustrating when you can't quite figure out how it all comes together.

The manuals include a user's guide (which also serves as a tutorial), a reference manual, and a "read-me first" booklet. An optional macro manual is available upon request. Among the "read-me" files is an introduction to Windows memory management that was the best we've seen in helping you understand the quirks of Windows and what you need to do to make Windows applications run better.

One of the features that really impressed us was mail-merge. Ami Professional's incarnation is the most flexible we have ever used. Unlike other word processors, Ami Pro lets you edit a document when you merge the variable fields. This allows you to tailor form letters, leaving the body of the letter pretty much the same while inserting and deleting sentences that customize the letter before you print the final form. After printing, you can use the same merge list to print out mailing labels — a feature we prefer to hand-inserting envelopes for printing.

The complete system comes on both 3.5- and 5.25-inch disks. You keep one and mail the other back in the prepaid mailer. Do make sure you get the latest

version, since a few minor but annoying bugs have been excoriated from the original release. System requirements are a 286- or 386-based machine, 640 Kbytes of memory, a hard disk drive, a mouse (optional but highly recommended), and DOS 3.0 or later. Be aware that spell checking is okay on a 386-based machine, but painfully slow on a 286-based one. Although a color system is not specified, we found it almost impossible to use the products on a computer equipped with an EGA-level plasma display. A runtime version of Windows comes with the package, but it doesn't give you all the features, like DDE, that Windows has to offer.

Our recommendation. At \$495 Ami Professional is competitively priced and an excellent value. Its what-you-see-is-what-you-get (WYSIWYG) capabilities combined with the Windows environment make it a versatile performer that serves the essential needs of both text-oriented users and graphics-oriented professionals. We remain impressed with Ami in both the standard and professional versions and recommend it highly to users looking to the future in word-processing software.

You can reach Samna Corp. at 5600 Glenridge Dr., Atlanta, GA 30342, phone (800) 831-9679.

Reader Service 21

Wordperfect enhancements

Wordperfect 5.1, the latest release of this popular word processor, includes a number of new and interesting features. The first noticeable difference shows up in the installation procedure. Much improved over previous versions, it offers provisions for installing virtually all of the different configurations of the program, including the network version. (There is no difference between the standard DOS version of Wordperfect 5.1 and the network version.)

The program arrived on 11 360-Kbyte diskettes. Since many of the files come in compressed format, you must use the installation procedure to decompress them.

Three of the 11 disks contained drivers for a wide range of available printers. If the driver for your printer is not supplied with the basic system, additional drivers are available at no charge. Additional font support is also available at a minimal cost for printers such as the HP Laserjet Series II.

Once you choose a printer, the instal-

lation program will provide a list of helpful hints and suggestions for using it. We installed the system with the following options enabled: the help system, spelling checker, thesaurus, graphics drivers, style library, keyboard files, graphics images, and learning files. Total disk space required was approximately 3.6 Mbytes. Eliminating the tutorial files in the Learn subdirectory will save you close to 256 Kbytes.

New features and enhancements.

The most significant enhancements to the Wordperfect interface are the introduction of pull-down menus and mouse support. In the default configuration, the

One of the most impressive new features, the equations editor, operates in graphics mode.

main menu does not initially appear on the screen but must be called forth either by pressing the right mouse button or using the Alt = keyboard combination.

Entries on the main menu include File, Edit, Search, Layout, Mark, Tools, Font, Graphics, and Help. A number of the entries lead to secondary menus. Often, once you choose an item, an additional menu appears — much like the choice menus used in Wordperfect 5.0. For example, selecting File from the main menu and then Print from the drop-down menu causes the familiar print options menu to appear. This same menu can be produced by hitting the Shift-F7 key combination, as in the previous version of Wordperfect.

New users, or those accustomed to pull-down menus, will appreciate the new menus feature. However, experienced users will find the menus offer little advantage over the function keys unless used with a mouse. Mouse use also speeds moving around inside a document or performing such functions as marking a block of text.

One of the most impressive new features, the equations editor, operates in graphics mode. It consists of a preview screen (distinct from the print preview screen), an editing window, and a set of palettes of standard symbols and text to include in your equations. The eight

separate palettes available include virtually any mathematical symbol or operator you might need.

You build equations into the editing window by selecting symbols from one of the palettes or typing any standard text. When you position the cursor over one of the symbols, the symbol's name appears on the screen status line. You can include this name in the equation, or you can include the graphics symbol itself. You can then redraw the equation on the preview screen and even edit it (within limits), such as moving the equation or scaling the display.

In addition to the extensive set of symbols, the equation editor offers a wide selection of commands for formatting your equation, such as aligning columns of a matrix or spacing symbols on a line.

Once you have created an equation, you can save it in its own disk file and later include it in your document.

If you have a graphics printer, Wordperfect can print any of the symbols found in the equation editor, regardless of the symbol's availability in your printer font set. It automatically prints symbols missing from your font set using bit-mapped graphics. We tried this feature with both an HP Laserjet Series II printer and an IBM dot matrix graphics printer. In both cases the system worked perfectly.

Since a good deal of our current work involves developing mathematical software, we found the equation editor a welcome addition to a sometimes limited set of mathematical documentation tools.

Another significant addition in Wordperfect 5.1, the tables feature, allows you to create two-dimensional tables of up to 32,765 rows and 32 columns, enclosed in a graphics outline. Each cell can contain text, numbers, or mathematical formulas operating on other cells, much like a standard spreadsheet. Virtually any format option available in a standard Wordperfect document, such as text justification and decimal point alignment, can be applied to a single cell or column of data.

The table editor allows you to treat a row or column of the table as a single entity. You can size a column, split it into multiple columns, move or copy either a column or row, and add or delete entire rows or columns. When printing a table, Wordperfect automatically adjusts the size of the table to fit within page margins. The table editor makes the construction and formatting of tables a simple process and offers a significant improvement over the old method of tab alignment to build tables.

Wordperfect 5.1 allows you to import spreadsheets from Planperfect, Lotus 1-2-3, or Microsoft Excel. The worksheet

is treated as a table with enough rows and columns automatically assigned to hold all of the data. If the worksheet contains too many columns to fit into a table, the maximum number of columns will be allocated and filled and the remainder of the spreadsheet lost. We imported a Lotus 1-2-3 spreadsheet with approximately 10 rows and 30 columns into a table and attempted to edit the contents of the table from the normal editing screen and also from within the table editor. In both cases we found the response time for even a simple cursor movement almost intolerably slow. However, for tables of moderate size contained on a single page, movement between different cells was not a problem.

The documentation package supplied with Wordperfect 5.1 is excellent and follows the same style as previous versions. However, the number of pages has grown substantially. The Wordperfect 5.0 manual contained approximately 480 pages, while the Wordperfect 5.1 manual now contains more than 1,000 pages. It has become so large that within a week the binder burst its seams. At this rate, we can expect Wordperfect 7.0 documentation to be available on CD-ROM only.

Summing up. We found each of the new features available in Wordperfect 5.1 to be an excellent addition to an already outstanding word processor. The equation editor and table editor together offer enough new capabilities to make an upgrade from Wordperfect 5.0 a worthwhile investment.

Wordperfect 5.1 is available from Wordperfect Corp., 1555 N. Technology Way, Orem, UT 84057. Wordperfect 5.1 operates in stand-alone mode or as a network server package. The retail price is \$495, with additional workstation packages available for \$295. If you currently own Wordperfect 5.0, you can upgrade to version 5.1 for \$85. Do not hesitate to do so.

Reader Service 22

A new Word, this time for Windows

Word for Windows is the Macintosh-like word processor for which we have been waiting ever since we put down our pencils and took up the keyboard as our primary ASCII writing tool. Well, no longer do we need to use archaic pre-computer tools. It's bit-mapped word processing for us from now on.

For anyone familiar with MacWrite and its successors, Word for Windows

(or WinWord, as it is becoming known) is an easy-to-learn, quick-to-use, powerful word processor that can meet varied users' needs, from first-time elementary school students right up to sophisticated office word-processing professionals. However, while it's easy to get started, proficiency with all of WinWord's features will take time and experience.

What is WinWord? Anyone who has used Windows and/or any of the Microsoft programming languages (or a Macintosh) will have no trouble getting started with WinWord and using all the standard Windows menu, mouse, and keyboard tools. As a basic word processor, WinWord provides a WYSIWYG data entry screen with all the standard word processor features. Beyond the essentials, WinWord also features style sheets and a macro programming language, just like Microsoft Word. The Basic-like macro language allows sophisticated users to create and implement powerful functions and features, including changing the pull-down menus and making your own click-boxes. While the complete macro language manual does not come with the numerous WinWord manuals, you can print out an abridged version using WinWord.

All is not peaches and cream with this first version of the product, however. While not sure that we would call some

The product comes with a ponderous and time-consuming on-line tutorial. But in checking it out, we discovered tips about mousing around that really improved our productivity.

of WinWord's idiosyncrasies "bugs," we do see some room for improvement. For example, if you don't have at least 500 Kbytes of RAM available, WinWord sometimes does funny things, like give you error messages that there isn't enough room to perform a function which it then proceeds to do just fine. Also, if you don't give this product the memory it demands, you will find on-

line help unavailable. All in all, though, we didn't find on-line help very useful. Considering Microsoft's wonderful work with hypertext-like help functions in their language products, we found this function a little disappointing.

WinWord features. Among WinWord's utilities are a thesaurus and a spelling checker. The spelling checker provides word meanings as well as spell checking. However, it is slow and misses words. The thesaurus is handy but also a little slow, making us wish we could access Word-for-Word, the really handy non-Windows thesaurus that has been our favorite for years.

Two attractive features are the icons used for the ribbon and the ruler. When enabled, these icons allow you to rapidly change fonts, font point sizes, and attributes (bold, underlined, italic, small caps, strikethrough, etc.) as well as character and paragraph styles, text alignment, tabs, and spacing. As a result, restructuring your document is both fast and easy.

Other features found in WinWord include a nine-level outline facility that allows you to set the level at which information is visible, table and graphics support, dynamic data exchange, annotations and revision marking, document templates, equations, and network support, just to name a few. All in all, WinWord is a compelling product that encourages you to explore ways to make better use of its facilities.

Documentation and on-line tutorial. WinWord's documentation is typical Microsoft. By this we mean lots of, of high quality materials. In fact, the user's reference is hardbound, a touch rarely if ever seen in microcomputer software packages. A problem with so much documentation is that it takes a lot of time to find the answer to a question. Also, we often felt unsure of the answer even after finally finding the relevant section.

The product comes with a ponderous and time-consuming on-line tutorial. But in checking it out, we discovered tips about mousing around that really improved our productivity. This surprised us, because we have used Macs and the like for a number of years now and Windows ever since it first became available. In addition to teaching us something new (what a surprise!), the tutorial really showed off some of WinWord's graphics and text features. Unfortunately, for reasons we cannot explain, the tutorial ended abruptly (bombed) part way through one of the lessons.

Getting started. Installation of the product's multitude of floppies (you get eight 1.2-Mbyte 5.25-inch and 14 3.5-

inch diskettes) is easy, if a little time-consuming. That wouldn't be bad, except that if you need to change a configuration feature, you must reinstall all the disks.

To run the system you will need a compatible 286 or 386-based PC, 640 Kbytes of memory, an EGA or VGA display adapter, and hard and floppy disk drives. A mouse is optional but very handy, and 1-2 Mbytes of EMS memory is highly recommended.

We tested WinWord on both a 12-MHz Toshiba T3200 and a PC Genius 386/20, each using a multisynch monitor with VGA graphics. From a performance standpoint, we don't think we would want to run this product at anything much less than 12 MHz. We did try to run the product with the T3200's regular EGA plasma display and did not consider that mode very useful. EGA resolution is probably all right, but we definitely recommend using a color display.

Features we liked best. We liked WinWord's print features best. Print Preview lets you see the shape and format of pages of your document as they would look printed. Of course, you can't read the microscopic text (unless you use a large point size), but you can get a feel for placement of headings, margins, page breaks, etc. In this mode, you can adjust all the page margins to get more or less of your document on a page, saving the time and expense of experimenting with your format on real paper.

Font selection too is very easy with this product. You can mix fonts at the character, word, sentence, etc. level and view the output directly as you type it in page mode. From a performance standpoint, however, the product's screen echo speed is really best when you use draft mode, the mode in which all text is displayed on the screen in one font. Also, page mode seems to use more memory,

thereby tempting the product to initiate one of the poor memory-error-handling routines mentioned above.

Of course, WinWord runs under Windows, so you have access to whatever printer and options you have selected with Windows' Control program. However, if you need to change a printer option, you can do that directly from within WinWord. That means you don't really need to run WinWord under Windows, but can use the Windows runtime module that accompanies the product. Using that module trims memory overhead but prevents you from using other Windows applications concurrently.

Document management with WinWord is really great. Whenever you save a new document, you can optionally save identifying information for later retrieval of documents whose names you have forgotten. In addition to a file name, you can give a document a title, description, and key words. When you need to find a file, you can search on any of those fields or on the basis of text strings within files.

WinWord comes with a number of file conversion utilities that let you import (and export) files from a variety of word processors and graphics packages. You can select the formats you plan to use when you install the product. These include Wordperfect, Multimate, several versions of ASCII text, and DCA/RTF. While not as extensive as that in Ami Professional, the list is sufficient for most users.

We did have a problem with importing ASCII files. WinWord provides conversion routines for a number of file types — character sets with or without the extended codes and with or without carriage return and line feed. However, WinWord uses the standard carriage-return code as its paragraph mark. This means if you import an ASCII file created by a word processor that inserts a

carriage return at the end of each line, WinWord treats each line like a separate paragraph. Hence, if your original file contains paragraphs composed of lines, each with one of these characters at the end, and you want to edit a line in a paragraph, wordwrap will not work for your entire paragraph, only for the line you are editing. However, we were able to quickly create a macro that allowed us to remove the carriage returns.

WinWord's multiple document handling capabilities (thanks to Windows) are also very good. If you have more than one document open at a time, an option lets you automatically optimize the display of all the documents, each in its own window. The resultant screen layout depends on the number of documents displayed. By simply clicking in a window, you can select the document to edit. Be warned, however, that these open documents use up more and more of WinWord's precious memory.

Conclusions. With the choice of several graphical word processors, users will have to consider which one to use based on particular features. WinWord is one of the most feature-laden of all those available, and it has more third-party support than the others as well. For example, you can pass electronic mail and send faxes from within WinWord. But this strength can also become WinWord's weakness, in that it can seem imposing at times. The new user can become intimidated by it all.

We rate WinWord a strong competitor for Ami and Wordperfect (described above). At \$495 it's both competitively and reasonably priced. You can't go wrong in choosing it, and you will find it an excellent buy. Contact Microsoft Corp. at 1 Microsoft Way, Redmond, WA 98052, phone (206) 882-8080.

Reader Service 23

Getting the most for your laser printer dollar

With the street price of laser printers now challenging the price of high-end dot matrix or ink jet printers, the price-performance ratio and the quality of laser printers make the competition look really terrible. Until we tested the Canon LBP-4 and LBP-8 Mark III laser printers, we had never needed to become terribly intimate with the jargon and complexities of the laser printer industry. But now we

have, and the results are terrific.

As we all know, if you want to squeeze every last drop of utility and performance out of your PC, you have to spend a lot more time at it than your loved ones would like. So it is with laser printers. It's just not as simple as switching the printer on-line and feeding in a bunch of paper. Now, you have to know about hard and soft fonts, pitch and point

size, minute differences among hundreds of available fonts, and the possible effects of your choices.

Canon laser printers

Both the Canon LBP-4 laser printer at \$1,545 and LBP-8 laser printer at

\$2,995, like their more expensive competitors, give you lots of features. The significant difference lies in convenience of use, weight and size, and printing speed. The LBP-4 weighs in at a light 23 pounds and, with a really small footprint of 14x16x8 inches, conveniently fits on most desktops. You can load paper from a front-loading paper bin or an optional cassette paper tray that fits under the printer. The cassette paper tray is so convenient, you might prefer it to opening the drop-down front loader for single-sheet feeding. The four-page-per-minute output can be directed either onto the top of the printer or toward the front. These input and output options make the printer environment quite compact.

The LBP-8 weighs a hefty 46 pounds and has a footprint of 21x18 inches. It is a serious office laser printer with a maximum output speed of eight pages per minute. You can load paper from the front-loading paper bin or the page tray that slides under the printer. Output can be directed either onto the top of the printer or toward the back onto a detachable tray. The variations in input and output options allow a lot of flexibility in setting up the printer. The LBP-8 also provides fairly quiet (43-53 decibels) operation.

Both laser printers are easy to assemble and use. It took us less than half an hour to unpack each printer and install the optional memory card, an optional font cartridge, the toner pack, and the paper bin. The factory default settings worked just fine for doing an off-line engine and factory-installed default font test as well as an on-line DOS print screen.

Both LBP printers' front panels consist of seven control switches used to set scores of features displayable on two half-inch-tall 16-character LCDs. We do admit that it took us a while to get the hang of the nested tree structure for option selection, but once we did we found you can set the laser to do almost anything from the control panel. A little explanation might be in order.

You can choose from six main option groups. The font/feeder lets you select and use a primary and secondary font from the eight built-in internal bit-mapped font sets or from up to two installed (optional) font cartridges. However, you cannot choose a scalable font from the front panel. These can be used only by application programs that support the printer. Nonetheless, for applications that don't recognize scalable fonts, this laser printer still works if you use a generic LBP-8 Canon print driver.

Other options let you select, among other things, the number of copies (up to 99), overlay of up to three images on one



Canon's LBP-4 laser beam printer

sheet of output, portrait or landscape mode (which runs a little slower), or parallel or serial interface (and possible settings). A macro function allows you to store a combination of instructions for later use. While the macros must be stored under computer control, you can execute them from the front panel. Indeed, 11 macros are built-in for setting things like horizontal tabs, page characteristics, and four forms (telephone memo, fax form, and two lined forms).

You can verify all switch-selected values by selecting the "test print" key. The values can then be stored (via the keys) in RAM or in NVRAM (nonvolatile RAM). You lose the contents of RAM when the machine is turned off, but the NVRAM settings are saved until you re-select the ROM defaults.

As you would expect, the LBP series of laser printers handles varied sizes of paper, transparencies, and labels. Envelopes feed in manually or from an optional envelope cassette. Options for both machines include interchangeable external font cartridges, an envelope cassette, and memory expansion from the standard 512 Kbytes to 2 Mbytes in the LBP-4 and from the standard 1.5 Mbytes up to 4.5 Mbytes in the LBP-8. The LBP-4 offers a built-in video interface with external connector.

Software support. Canon laser printers are not supported by as many packages as are other, HP-compatible, laser printers. However, the list of popular ap-

plications that can take advantage of the machines' rich function is growing. And, as we've mentioned, the LBP-4 is compatible with the LBP-8, which is included in most sets of application-specific device drivers.

This printer comes with a set of Canon-supplied device drivers. The applications supported include Multimate, GEM, Lotus, Windows, Word, OfficeWriter, Q&A, Wordperfect, Wordstar, and Xywrite. If your software isn't among these packages but supports an HP laser printer, see our accompanying review of Laserwin from Metro Software ("Practical laser utilities" below).

Documentation. Both machines come with an operator's manual. The one for the LBP-4 is a little less comprehensive, but both manuals contain sufficient information to explain all you need to know about installing and running your new printer. The more than a dozen chapters are indexed and well illustrated, for a result that is thorough, easy to follow, easy to understand, and replete with step-by-step examples. Additional manuals come with the font cartridges, the RAM expansion memory, and the paper cassette feeder.

The one missing piece in all of this is a manual explaining the CaPSL (Canon Printing System Language) controller. If you really want to take advantage of all the unusual features that scalable fonts offer, you will need to understand CaPSL. It draws any shade of gray,



Canon's LBP-8 Mark III printers

places fonts in any orientation, and offers a number of plotting commands for drawing vectors, among other features. It also implements a set of the ANSI/ISO screen control commands and can handle both bit-mapped and scalable fonts. Like Postscript, it accepts outline-font "hints" and has macros with some programming language features. And it features speed optimization, like marker drawing commands and image compression. But you won't know all this unless you purchase the expensive, optional manual sets, which retail for \$120.

Customer service. If you are going to pay more and get a device as complex as a laser printer, you had better be sure you can get your questions answered. We guarantee that you will have questions.

Canon provides toll-free access to their New York-based customer service department. And that service is just great. The service technicians know about all the Canon printer products and can guide you through step-by-step diagnoses of really arcane issues. Besides handling product-specific issues, Canon tech support educated us on font selection and printer software support. Canon's customer service department is reason enough to select a Canon product for your first laser printer.

A visit to the local Canon product support office produced the same high-quality treatment that we had received over the phone. We visited the office because

our LBP-8 refused to print anything when first unpacked. After reading the manual without finding a cure, we called the hotline. The service rep suggested that our problem might be with the toner cartridge. Canon service personnel at the local product support office examined the printer while we waited, then replaced the toner cartridge.

The warranty that comes with the LBP lasers is good for one year. Standard warranties for other popular laser printers generally run for 30-90 days.

Summing up. Based on both price and performance, we place these laser printers at the top of our list. And that should be no surprise. Since the Canon engine is widely used, you would expect the company's own products to be superior. They are. Add the excellent customer service and you have very good reasons to purchase a Canon LBP printer.

If you are in the market for a professional quality printer, either for a large office or simply for your own small business, these Canon products come highly recommended. And Canon has just sweetened the deal. Until June 30, 1990 you can get the SC-1 IC card with 22 more scalable fonts in seven typefaces free when you buy an LBP printer.

Canon USA, Printer Division, is located at 1 Canon Plaza, Lake Success, NY 11042, phone (516) 488-6700.

Postscript printing

Having seen what a Postscript-compatible printer can do, many of us have wished we could afford one. The Postscript page description language (PDL) provides a wide range of typeset graphics capabilities, including a multitude of scalable fonts, the ability to rotate and translate images, and excellent control over shades of gray as well as color. Applications that support Postscript produce the best-looking output when connected to a Postscript-compatible printer. The main reason that most of us don't own a Postscript-compatible printer is that they cost significantly more than traditional laser printers.

Postscript interpretation software substitutes the processing power of your PC for the embedded microprocessor within a Postscript-compatible printer as the means for interpreting Postscript commands. These resident printer utilities capture and interpret Postscript output from an application and construct the output page image to be printed. This approach saves the user lots of money. Typically, Postscript compatibility costs between \$200 and \$500 dollars, with the difference in price related to the number of typefaces supported.

Of course, what you save in dollars you pay in time and space. First, you can't offload the Postscript file to a print engine and continue processing on your PC — your PC is the print engine. And, unless you have a fast 80386-based PC, it will take some time to process the Postscript file (normally done by a Motorola 680x0 microprocessor embedded in a laser printer). Second, you need a significant amount (1-2 Mbytes) of extended memory beyond the standard 640 Kbytes in which to generate the bit-mapped images from the Postscript files.

In spite of these constraints, you get the ability to print Postscript documents on a wide variety of dot matrix, ink jet, and laser printers. Considering that you can buy a speedy 24-pin dot matrix printer for \$300 or a four-page-per-minute laser printer for less than \$1,000 and add on a Postscript compatibility package for another \$200, you can get advanced text and graphics printing capability at an affordable price. Additionally, Postscript emulators typically allow you to

- print from within normal applications as if you were printing to actual Postscript printers,
- spool Postscript files to disk for later printing,
- preview Postscript output, and
- concurrently send print files to

Reader Service 24

virtual LPT ports for Postscript files and real ports for normal files.

We had the opportunity to review two excellent packages for the PC, Prescript from Pan Overseas Computer and Ultrascript from PC from QMS. Each well-done product offers basic Postscript compatibility, plus extra features that might prove useful in determining which product you should consider purchasing.

Prescript, Version 1.1. Pan Overseas Computer is best known for its line of power-safe business computers with built-in uninterruptible power supplies. The company also offers desktop publishing software packages, among them its Prescript products. The two versions available, Prescript-Standard and Prescript-Deluxe, retail for \$195 and \$395, respectively. Prescript-Standard comes with 13 typefaces, while Prescript-Deluxe provides a total of 35 typefaces. The 13 standard typefaces are Courier, Helvetica, and Times Roman in bold, oblique, and bold-oblique, plus Symbol. This provides good variety at a low entry price.

Prescript requires an 80286 or 80386 PC AT-compatible computer, plus 2 Mbytes of extended memory and 2.5 Mbytes of hard disk space for the program and the fonts. It supports CGA, Hercules, EGA, VGA, or Super VGA display adapters, and higher resolution monitors that have appropriate Windows drivers. It runs on DOS 3.1 or higher as a protected-mode terminate-and-stay-resident program. It conforms to the virtual control program interface (VCPI) specification for protected-mode programs running under DOS.

You can use Prescript with the HP Laserjet, HP Series II, and HP Series IIP laser printers, and all 100 percent HP-Laserjet (PCL) compatibles. These printers require 1 Mbyte of printer memory for full-page graphics. Prescript also works with the Deskjet and Deskjet Plus ink jet printers because they are PCL-compatible printers. A numeric coprocessor chip will speed things up but isn't required.

Prescript works with any Postscript application, such as Pagemaker, Ventura Publisher, Coreldraw, Wordperfect, Excel, Word, Ami, and on and on. Although it runs under Windows/286, it will not run under Windows/386 (due to the VCPI conflict between DOS extenders and Windows). To print files from applications created within Windows/386, you must save the document, exit Windows/386, and print the file using Prescript at the DOS level.

Prescript is easy to install using the setup procedure, which unpacks the application code. Once loaded, it operates

as transparently as if you had a real Postscript printer. The thorough and fully indexed manual that comes with the software describes how to install it in a wide variety of applications software. We conducted our tests using it with Windows/286, Pagemaker, Coreldraw, and Wordperfect. The print quality is quite remarkable, and the package provides an extremely economical way for you to get into Postscript printing. Published reports have indicated that Prescript is among the fastest of the interpreters, but we took no performance measurements.

Contact Pan Overseas Technologies at 44 Route 46, Pine Brook, NJ 07058, phone (201) 808-1900, fax (201) 808-9889.

Reader Service 25

Ultrascript PC, Version 2.0. QMS is well known as a supplier of Postscript printers. Thus, it comes as no surprise that they offer a software interpreter that uses the same licensed typefaces as Adobe, the company formed to market Postscript. The two versions available, Ultrascript PC and Ultrascript PC Plus, retail for \$195 and \$445, respectively. The Plus version adds an additional 22 typefaces to the base set of 25 that comes with the lower priced version. Of course, if you buy Ultrascript PC, you can add the additional typefaces later from QMS's extensive type collection. Since most of us can live with Courier, Helvetica, Lucida, Lucida Sans, Lucida Sans Typewriter, and Times Roman in regular, bold, italic, and bold italic, plus Symbol, the lower priced package is the one to start with.

Like Prescript, Ultrascript is easily installed using the setup procedure, which unpacks the software from either the 3.5-inch or 5.25-inch disks supplied in the package. You will need at least 4 Mbytes of hard disk space to hold the executables along with the numerous typefaces. Also, you'll need a PC AT or 386 machine with 640 Kbytes of memory and running DOS 3.1 or higher. A numeric coprocessor chip will speed things up but isn't required. To print from within an application, you'll need 1.5 Mbytes of extended (not expanded) memory. Some of the setup questions relate to how you want to use the program (from within an application or stand-alone). After answering these questions, you next run and configure the system to the type of printer you're using, the resolution (draft or final), and how the printer is physically connected.

You can run Ultrascript in a variety of ways. Like Prescript, it runs under Microsoft Windows/286 but not Windows/386 due to the VCPI conflict. If

you have only 640 Kbytes of memory, you can use the "lightweight" version to print Postscript files. Of course, you can load Ultrascript permanently before you execute your Postscript application so that it operates as transparently as if you had a real Postscript printer. Using the Capture TSR utility, you can redirect a nonexistent printer port (say LPT3) to the Postscript interpreter so that it appears as if you have both a normal and a Postscript printer.

One of the features that sets Ultrascript apart from some of the other packages like it is its support for a wide variety of printers, including the HP Laserjet Series II and IIP; HP Deskjet and Paintjet; Canon LBP-4 and LBP-8II; Canon Bubblejet 130; Epson LQ, LX, and FX series of dot matrix printers; IBM Proprinter and Graphics Printer; and NEC Pinwriter. Another feature is the ability to save documents as TIFF and PCX files. This latest version also supports color Postscript. And finally, a convenient DOS interface features a series of pull-down menus to configure the software and hardware, print files from any subdirectory, and interpret Postscript commands entered directly from the terminal. The last feature in particular helps you learn the Postscript language.

The manual that comes with the software is thorough and fully indexed. It covers all the details from installing to using the software with one exception: it never discusses the various windows in the DOS interface. That interface is helpful in that it provides three windows for displaying the status, the Postscript output, and the help messages. While not explained, that interface is fairly intuitive with the exception of certain fields that don't seem to relate to any of the system commands.

As mentioned earlier, Postscript interpretation is time-consuming but not unreasonably so. When we used Ultrascript to print on an Epson LQ1050, the print speed was quite acceptable, on a par with typical graphic printing to that device. Output to a Kyocera F-1000A and a Canon LBP-4 was slower, but the visual results were far more acceptable. What really amazed us was that the results for each printer looked practically identical (ignoring the differences in resolution between devices).

We made comparisons to an Apple Laserwriter by sending the same Postscript file to it as well as to the Kyocera. To the naked eye the results were identical, but print times differed noticeably.

We recommend Ultrascript as an excellent buy, offering flexibility and performance for users who don't have Postscript printers and need to print Postscript documents. Ultrascript is available from

Imagen Corp., 2650 San Tomas Expressway, PO Box 58101, Santa Clara, CA 95052-8101, phone (408) 986-9400, fax (408) 727-3725.

Reader Service 26

Practical laser utilities

Metro Software offers a number of interesting and extremely useful utilities under the product names of Laserwin and Superfonts 25/1. In one of its incarnations, Laserwin acts as an enhanced HP Laserjet Plus emulator for the Canon LBP-8 Mark II/III and the LBP-4 (discussed above). In another, it fools your ink jet or dot matrix printer into thinking that it is an HP Series II printer. In either case, since the basic HP printer only includes a few standard fonts, the Superfonts utility gives you all of the Hewlett Packard cartridges from A-Z. So, for less than \$200 for each product, you can completely change the way you print.

Understanding Laserwin. Installing either version of Laserwin is very easy. Because the package loads itself as a TSR, it acts as a resident translator that accepts HP laser output directed to a printer port, reformats it to the form required for your printer, and produces Laserjet-like quality output. The cost to you is a small amount of memory (27 Kbytes) and the time it takes to make the translation. Gaining the ability to run applications that you could not before because they required a Laserjet printer may make the effort worthwhile.

Once installed in memory with the LT command, Laserwin uses a fairly simple command set (of the form "LT <parameter>") to control what you can do next. For instance, the H parameter gives you a help screen. C# and L# override the printer port you are using. The I and A commands make the printer inactive or active; in the inactive mode, printer output is not translated. The T parameter tells Laserwin to expand all fonts downloaded to the printer, while the K parameter keeps the expanded fonts so they need not be expanded each time they are downloaded. Other parameters invoke a print-to-disk feature, change the default paper size, reset the printer, send a form-feed, offer status information, and unload Laserwin from memory.

The ability to alter parameters at the command line in a commonsense fashion is really quite handy. One reason you might want to do so is that using Laserwin can take a lot of CPU cycles — it's slow: Because of that, you might want to turn some of the options on and off to optimize your print time.

If you used Laserwin simply as an HP Laserjet translator, you wouldn't need to know any more than this. It would all work transparently, and the output would really please you. If you need to know more, Metro's excellent manual offers a section explaining Laserjet Plus commands — a real benefit if you want to understand laser printer escape sequences. We only wish they had done the same for the CaPSL language supported in their Canon laser version.

What if you want to use this utility with software that doesn't support the Laserjet Plus? Metro has conveniently designed its own embedded text commands for insertion into any output to be printed. The format of the command is [xxx where xxx stands for a three-letter abbreviation such as UND to toggle underlining, REV to toggle reverse video out (that is, white on black), F80 to select a built-in font, or FOR for form-feed. This small sample only touches on the multitude of commands that Laserwin has to offer, but it gives a sense of how easy it is to embed the commands in plain text as well as within most word processors so the results are passed directly to the translator.

Fonts available include 12 "quick fonts" representing the 8- and 12-point portrait and landscape fonts built into the real Laserjet Plus. Using Superfonts, you can download additional fonts and tell Laserwin which one to use in the future by selecting the font according to its characteristics (point size, pitch, portrait/landscape, italics/upright, stroke weight, and symbol set).

System requirements for Laserwin are a compatible PC running DOS 2.0 or higher, one floppy drive and a hard drive, and a modest amount of memory (unspecified). The software supports dot matrix and ink jet printers from more than 15 different manufacturers. The retail price for Laserwin is \$179 (for dot matrix, ink jet, and Canon printers), with an introductory price of \$149 (for dot matrix and ink jet printers).

Understanding Superfonts 25/1.

About the time you get the hang of using Laserwin, you begin to realize that the Laserjet Plus sorely lacks internal fonts. That's when you also realize that the ability to insert one of the HP cartridge fonts into your pseudo Laserjet printer would be very handy. With a pseudo Laserjet you need a pseudo cartridge to insert. You need Superfonts 25/1. Like its electrical counterpart, a cartridge containing all 25 fonts in one package, Superfonts allows you to generate and download any number of equivalent HP font cartridges into your emulated Laserjet printer.

Since these fonts take up disk space, they are kept in a compressed form. When you call up Superfonts, you get a display screen from which to select the fonts you want to "make." Selecting a font tells you how much memory it will occupy when you generate it (it can be considerable). Making it puts it on your hard disk for later downloading using the Insert utility.

Another choice in the main menu lets you change the colors of the screens. The last choice in the menu is Quit. Once you leave Superfonts, you see a closing screen that explains the Insert, Remove, Whatis, and Sample commands that you use to manage your cartridge fonts. Insert and Remove are obvious. Whatis displays the contents of a cartridge, while Sample prints out a sample.

System requirements for Superfonts are the same as for Laserwin. Installation is from an install file that loads the six floppies the software comes on. Obviously, Superfonts works with an HP Laserjet Plus printer, but that isn't what we tested it with in this review (see below). The retail price is \$179 and the introductory price \$149.

Using and evaluating Laserwin.

Laserwin works as advertised. We tested one of the Laserwin products with a Canon BJ-130 ink jet printer and the other Laserwin with both a Canon LBP-4 and a Canon LBP-8 laser printer using a 12-MHz Toshiba T3200 and a Compaq Deskpro 286. In all cases we found that the Laserwin products could appreciably improve the quality of the output. However, the length of time it took for output to print was unacceptably long for the inkjet printer, even with the spooler installed on a RAM disk and background printing selected as an option.

On the other hand, results with the Canon LBP printers took less time. And, when combined with Superfonts, Laserwin allowed us to print on the LBPs in a variety of fonts and orientations that ordinarily are difficult to do. Canon provides drivers that allow you to use the LBP series and its built-in scalable fonts with a meaningful, but limited, number of software packages. If you don't use these packages, you find yourself with a rather sophisticated printer and no way to use all of its features.

So, if you want HP emulation for occasional high-quality output, and you only have a dot matrix or bubble jet printer, then Laserwin provides acceptable results. If you want to substitute your LBP-4 or -8 for the HP Laserjet, Laserwin is mandatory.

The manuals for the Laserwin products and for Superfonts are well written and thorough, although obviously miss-

ing a section needed to explain how to use Superfont with Lasertwin. This omission caused us some grief while trying to figure out which to load first, how to download fonts, and if it was even possible to select Superfont by font number (it isn't). A call to customer service resulted in the company's agreeing to add this information to future manuals — a necessary addition.

Metro Software has a scheme that requires you to call them during installation (on an 800 number) to get your soft-

ware registered. Even if you don't phone, you'll still be able to install and copy the product — for backup, of course! Technical support was very responsive both during that call and later, when they returned other calls regarding the questions we had about using their products.

We felt this software delivers what it promises. We could only find one example where it failed to act in a 100 percent compatible fashion (with 4print from Korenth Associates). What makes these utilities so valuable, however, is

that they offer HP compatibility at a reasonable price.

Around the time this review appears, a new Canon version will provide HP Laserjet II compatibility, as well as scalable font support and a print spooler. Contact Metro Software at 1870 W. Prince Rd., Suite 70, Tucson, AZ 85705, phone (602) 292-0313 or (800) 621-1137, fax (602) 292-1563.

Lasertwin: Reader Service 27
Superfont: Reader Service 28

Review notes

A command processor for MS-DOS, Command Plus (Cplus) from ESP Software Systems, 6140 Bristol Parkway, Culver City, CA 90230, phone (213) 645-5810, fax (213) 645-5810 is essentially a command language interpreter that confers Unix-type capabilities to MS-DOS. From this point of view, it's an important enhancement. It translates commands typed at the keyboard into system actions.

Features include compatibility with current DOS commands (including DOS 4.0), significant strengthening of the standard DOS commands (such as Dir, Copy, or Del), support for aliases, command history, a Script capability, and a block-structured language for batch processing. The batch processing language supports integer and string variables and includes a library of routines.

The aliasing mechanism can transform commands. The history facility retains up to 125 commands, and commands can be reexecuted based on their number in the history buffer or on a prefix of their names. A "control panel" command allows you to examine the content of the memory at any moment and to inspect various state variables. Another useful feature, the keyboard macro processor, allows attachment of commands to specific keys.

A "file viewer" (Browse) displays the content of files (in 43- or 50-line modes) and supports regular expression string searches. Finally, I should mention the screen-saver and a Log

processor that lets you track system usage. The documentation is excellent, with many examples and detailed explanation of command options.

Cplus operates on most IBM XT, AT, PS/2, or compatible PCs. It comes on 5.25-inch or 3.5-inch floppies and costs \$159.95.

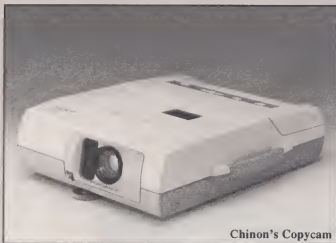
All these facilities come at a price. In the absence of expanded memory, Cplus occupies about 17 Kbytes of the lower

Recovering meeting presentations. Copycam, a portable image copier, is an interesting device from Chinon that "photographs" chalk boards or easels so you can have hard copies of what you create in typical meetings. Disguised in the shape of a 35 mm projector, this copier uses a digital camera to generate an instant copy of anything within its field of view (typically 1.2m to infinity). Unlike "electronic" blackboards, it uses no large and bulky special writing surface. The 11-pound package is easily moved and quickly set up. The few controls include a selection key (blackboard, whiteboard, or scenery), copy style (normal or enlarged), contrast, and start.

At \$1,495 this isn't an inexpensive item, but it sure is handy. I used it as a point-and-shoot device, picking up any stationary object. An optional NiCad battery backup is available with a claimed capability of more than 100 pages of copy on one charge. Printing is thermal, and a full page starts rolling out immediately after you press the start button. Print times are on the order of 20 seconds.

This handy device is just right for corporate meetings, particularly brain-storming sessions where there isn't time to plan ahead to provide hard copies of presentation materials. For more information, contact Chinon America, 660 Maple Ave., Torrance, CA 90503, phone (800) 441-0222 or (213) 533-0274.

— R. Eckhouse



Chinon's Copycam

ranks of memory. Annoying problems might appear, especially when you use large database systems with their own demands on memory resources. Unix users will soon discover that certain well-known commands, though present, have rather different properties and limitations. For instance, though Cplus supports a directory stack (limited to five entries), the commands governing the stack, "pushd" and "popd," behave differently compared to their Unix counterparts.

Nevertheless, Cplus can provide a helpful addition to DOS-based systems.

— D. Simovici

ADVERTISER INDEX

Amdahl Corp.	22
CACI Products Co.	1
CAIA '91	78
EDAC '91	64
Elsevier Science Publishing Co. Inc.	19
GTE Laboratories, Inc.	21
IEEE Computer Society Membership	34, 77, 86, 122
IEEE Computer Society Ombudsman	81
IMS '91	77
Learning Tree International	53
Micro Way	Cover IV
Oakland Group, Inc.	4
Parallel Processing Conference	102-103
Scientific Engineering Software, Inc.	Cover II
The Swiss Federal Institute of Technology (ETH Zurich)	81
University of Minnesota, Department of Computer Science	122
USENIX Technical Conference & Exhibition	8
Water Scale Integration Conference	Cover III
Xerox Corp.	7
Classified Advertising	118-122

FOR DISPLAY ADVERTISING INFORMATION, CONTACT:

Northern California and Pacific Northwest: Roy McDonald Assoc. Inc., 5915 Hollis St., Emeryville, CA 94608; (415) 653-2122.

Jim Olsen, P.O. Box 696, Hillsboro, OR 97123; (503) 640-2011.

Southern California and Mountain States: Richard C. Faust Co., 24050 Madison St., Suite 100, Torrance, CA 90505; (213) 373-9604.

Midwest: The Kingwill Company, 4433 W. Touhy Ave., #540, Lincolnwood, IL 60464; (708) 675-5755.

East Coast: Atlantic Representative Group, 349 Maple Pl., Keyport, NJ 07735; (201) 739-1444.

New England: Arpin Associates, P.O. Box 6444, Holliston, MA 01746; (508) 429-8907.

Europe: Heinz J. Görgens, Parkstrasse 8a, D-4054 Nettetal 1 - Hinsbeck (F.R.G.); phone: (0 21 53) 8 99 88; telex 841(17)2153310-HJG ttx d.

Southwest/Southeast: Heidi Rex, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380.

For production information, conference, and classified advertising, contact Heidi Rex or Marlan Tibayan.

COMPUTER, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; phone (714) 821-8380; fax (714) 821-4010.

PRODUCT INDEX

	RS#	PG#
Aeon Systems	120	100
Amdahl Corp.	—	22
Apple Computer, Inc.	30	97
Ariel Corp.	121	100
AST Research	122	100
C&C Technology	124	100
Cache Computers	123	100
CACI Products Co.	—	1
CAIA '91	—	78
California Scientific Software	39	99
Canon USA	24	92
Classified Advertising	—	118-122
Compaq Computer	125	100
Data General	127	100
Datacom	126	100
Datapro	128	100
Dedicated Computer Systems	129	100
Dell Computer	130-131	100
Digital Equipment Corp.	31-32	97
Digital Optics Ltd.	36	98
Dolphin Scientific	132	100
EDAC '91	—	64
Elsevier Science Publishing Co. Inc.	3	19
Encore Computer Corp.	40	99
GTE Laboratories, Inc.	—	21
Hitachi America	139	101
IEEE CS Membership	—	34, 77, 86, 122
IEEE CS Ombudsman	—	81
Imagem Corp.	26	94
IMS '91	—	77
Industrial CPU Systems	133	101
Integrated Device Technology	138	101
IntellCorp, Inc.	37	99
Learning Tree International	4	53
LSI Logic	140	101
Metro Software	27-28	95
Micro Way	5, 134	101, C.4
Microsoft Corp.	23	90
Motorola	141	101
National Semiconductor	142	101
Oakland Group, Inc.	2	4
Otec Technologies	135	101
Pan Overseas Technologies	25	93
Parallel Processing Conference	—	102-103
Sanna Corp.	21	88
Scientific Engineering Software, Inc.	1	C.3
Sharp Electronics	136	101
Silicon Graphics, Inc.	38	99
Spectrafax	35	98
Swiss Federal Institute of Technology (ETH Zurich)	—	81
Unisys Corp.	33	97
University of Minnesota,	—	122
Dept. of Computer Science	—	8
USENIX Technical Conference	—	8
ULSI Technology	143	101
Water Scale Integration Conference	—	C.3
Wang Laboratories, Inc.	34	98
Wonderful Corp.	22	89
Xerox Corp.	—	7
Yarc Systems	137	101

COMPUTER

Reader Service Card

May 1990 issue (Card valid after October 1990)

Print address below.

Name _____

Title _____

Company _____

Address _____

City _____

State/ZIP _____

Country _____ Phone _____

Please send (Circle those you want)

- | | |
|----------------------------|---------------------------------------|
| 201 Publications catalog | 203 Student membership application |
| 202 Membership information | 204 Application for senior membership |

Reader interest

(Circle what you liked, add comments on the back)

- A Guest Editor
- B Distributed File Access
- C x-kernel
- D Mach Operating System
- E Amoeba
- F Shared Memory
- G Hierarchical Control
- H Standards
- J CS News
- K Product Reviews
- L New Products
- M ICs/Microsystems
- N Conferences
- P Calendar/Call
- Q Book Reviews/New Lit
- R CS Magazines
- S Open Channel

Product information

(Circle numbers for products and advertisements on which you want more information)

1	21	41	61	81	101	121	141	161	181
2	22	42	62	82	102	122	142	162	182
3	23	43	63	83	103	123	143	163	183
4	24	44	64	84	104	124	144	164	184
5	25	45	65	85	105	125	145	165	185
6	26	46	66	86	106	126	146	166	186
7	27	47	67	87	107	127	147	167	187
8	28	48	68	88	108	128	148	168	188
9	29	49	69	89	109	129	149	169	189
10	30	50	70	90	110	130	150	170	190
11	31	51	71	91	111	131	151	171	191
12	32	52	72	92	112	132	152	172	192
13	33	53	73	93	113	133	153	173	193
14	34	54	74	94	114	134	154	174	194
15	35	55	75	95	115	135	155	175	195
16	36	56	76	96	116	136	156	176	196
17	37	57	77	97	117	137	157	177	197
18	38	58	78	98	118	138	158	178	198
19	39	59	79	99	119	139	159	179	199
20	40	60	80	100	120	140	160	180	200

COMPUTER

Reader Service Card

May 1990 issue (Card valid after October 1990)

Print address below.

Name _____

Title _____

Company _____

Address _____

City _____

State/ZIP _____

Country _____ Phone _____

Please send (Circle those you want)

- | | |
|----------------------------|---------------------------------------|
| 201 Publications catalog | 203 Student membership application |
| 202 Membership information | 204 Application for senior membership |

Reader interest

(Circle what you liked, add comments on the back)

- A Guest Editor
- B Distributed File Access
- C x-kernel
- D Mach Operating System
- E Amoeba
- F Shared Memory
- G Hierarchical Control
- H Standards
- J CS News
- K Product Reviews
- L New Products
- M ICs/Microsystems
- N Conferences
- P Calendar/Call
- Q Book Reviews/New Lit
- R CS Magazines
- S Open Channel

Product information

(Circle numbers for products and advertisements on which you want more information)

1	21	41	61	81	101	121	141	161	181
2	22	42	62	82	102	122	142	162	182
3	23	43	63	83	103	123	143	163	183
4	24	44	64	84	104	124	144	164	184
5	25	45	65	85	105	125	145	165	185
6	26	46	66	86	106	126	146	166	186
7	27	47	67	87	107	127	147	167	187
8	28	48	68	88	108	128	148	168	188
9	29	49	69	89	109	129	149	169	189
10	30	50	70	90	110	130	150	170	190
11	31	51	71	91	111	131	151	171	191
12	32	52	72	92	112	132	152	172	192
13	33	53	73	93	113	133	153	173	193
14	34	54	74	94	114	134	154	174	194
15	35	55	75	95	115	135	155	175	195
16	36	56	76	96	116	136	156	176	196
17	37	57	77	97	117	137	157	177	197
18	38	58	78	98	118	138	158	178	198
19	39	59	79	99	119	139	159	179	199
20	40	60	80	100	120	140	160	180	200

COMPUTER

Reader Service Card

May 1990 issue (Card valid after October 1990)

Print address below.

Name _____

Title _____

Company _____

Address _____

City _____

State/ZIP _____

Country _____ Phone _____

Please send (Circle those you want)

- | | |
|----------------------------|---------------------------------------|
| 201 Publications catalog | 203 Student membership application |
| 202 Membership information | 204 Application for senior membership |

Reader interest

(Circle what you liked, add comments on the back)

- A Guest Editor
- B Distributed File Access
- C x-kernel
- D Mach Operating System
- E Amoeba
- F Shared Memory
- G Hierarchical Control
- H Standards
- J CS News
- K Product Reviews
- L New Products
- M ICs/Microsystems
- N Conferences
- P Calendar/Call
- Q Book Reviews/New Lit
- R CS Magazines
- S Open Channel

Product information

(Circle numbers for products and advertisements on which you want more information)

1	21	41	61	81	101	121	141	161	181
2	22	42	62	82	102	122	142	162	182
3	23	43	63	83	103	123	143	163	183
4	24	44	64	84	104	124	144	164	184
5	25	45	65	85	105	125	145	165	185
6	26	46	66	86	106	126	146	166	186
7	27	47	67	87	107	127	147	167	187
8	28	48	68	88	108	128	148	168	188
9	29	49	69	89	109	129	149	169	189
10	30	50	70	90	110	130	150	170	190
11	31	51	71	91	111	131	151	171	191
12	32	52	72	92	112	132	152	172	192
13	33	53	73	93	113	133	153	173	193
14	34	54	74	94	114	134	154	174	194
15	35	55	75	95	115	135	155	175	195
16	36	56	76	96	116	136	156	176	196
17	37	57	77	97	117	137	157	177	197
18	38	58	78	98	118	138	158	178	198
19	39	59	79	99	119	139	159	179	199
20	40	60	80	100	120	140	160	180	200

Editorial comments

I liked: _____

I disliked: _____

I would like to see: _____

For reader-service inquiries, see other side.

PO Box is for reader-service cards only.

PLACE
POSTAGE
HERE

COMPUTER

Reader Service Inquiries

PO Box 16508

North Hollywood, CA 91615-6508

USA



Editorial comments

I liked: _____

I disliked: _____

I would like to see: _____

For reader-service inquiries, see other side.

PO Box is for reader-service cards only.

PLACE
POSTAGE
HERE

COMPUTER

Reader Service Inquiries

PO Box 16508

North Hollywood, CA 91615-6508

USA



Editorial comments

I liked: _____

I disliked: _____

I would like to see: _____

For reader-service inquiries, see other side.

PO Box is for reader-service cards only.

PLACE
POSTAGE
HERE

COMPUTER

Reader Service Inquiries

PO Box 16508

North Hollywood, CA 91615-6508

USA



NEW PRODUCTS

Contact or send releases to Nancy Hays, Computer, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; Compmail+, n.hays

Apple announces its fastest Mac yet

Apple Computer calls the new Macintosh IIx the fastest Mac yet developed. The Mac IIx uses a 40-MHz 68030 microprocessor and a 68882 floating-point coprocessor. It features a built-in 32-Kbyte static RAM cache memory subsystem, latched-write design, two dedicated I/O processors, a SCSI/DMA controller, six Nubus slots, and an expansion slot tied directly to the processor.

The new PC uses Apple's 1.4-Mbyte Superdrive disk drive and will accommodate a second Superdrive. According to the company, the Mac IIx also has improved design elements, such as a quieter, variable-speed fan and an adjustable extended keyboard.

The Mac IIx provides full ROM support for Appletalk protocols and includes

serial ports for LocalTalk connections.

The Macintosh IIx with 4 Mbytes of memory costs \$8,969 for the Superdrive version, \$9,869 for the 80-Mbyte internal hard disk drive version, and \$10,969 for the 160-Mbyte internal hard disk drive version. Standard features include a mouse, Macintosh ports, Hypercard 1.2.5 software, and System Software 6.0.5. The software comes preinstalled on hard disk configurations. A/UX Version 2.0 is optional.

Macintosh II and IIx customers can upgrade to a Mac IIx with a logic board upgrade kit costing \$2,999. A 4-Mbyte memory expansion kit for the Mac IIx costs \$999.

Reader Service 30



Apple Computer's Macintosh IIx incorporates a 40-MHz 68030 microprocessor and 68882 coprocessor to become what the company calls its fastest Mac yet.

DEC debuts design, database tools for VAX/VMS

Digital Equipment has announced two new software tools: DECtrace and RdbExpert.

DECtrace for VMS reportedly provides a mechanism for collecting and reporting event-based data and performance information from applications and databases. According to the company, DECtrace event data can be gathered from any combination of VMS layered products and application programs containing DECtrace service routine calls. (Users can add service routine calls to their own applications.)

DECtrace operates in single VAX systems and VAXcluster modes. It is integrated with VAX Rdb/VMS, VAX DBMS, and VAX ACS and provides similar support for DECintact and non-TP monitor applications through the 3GL interface. It can provide detailed database transaction workload information for RdbExpert. It formats data collection files into an Rdb/VMS database or RMS file for query by user programs or 4GLs.

DECtrace users can specify cluster-wide or local node data collection, plus time, duration, classes of data, and storage. They can also cancel data collection. Reports generated are Detail, Sum-

mary, and Frequency.

Shipments of DECtrace are scheduled for June. Digital has established a base price of \$3,427.

RdbExpert, a VMS-based product, generates VAX Rdb/VMS and VAX DBMS physical database designs. According to Digital, the software analyzes the database logical design, the transaction workload information, the data volume information, and the system environment. It then generates executable procedures for creating the physical database design.

RdbExpert includes the VAX Rdb/VMS query optimizer. It provides DECwindows and command-line interface facilities. It can import database information and DECtrace workload data over the network.

The base price for RdbExpert is \$12,750, with shipments scheduled for June. Development versions of VAX Rdb/VMS and VAX DBMS must be available to execute the commands to create or re-create the optimized databases. The development versions can reside on a system other than the one on which RdbExpert is executed.

DECtrace: Reader Service 31
RdbExpert: Reader Service 32

New Unisys A-Series models powered by Scamp

Unisys has announced new models in its A-Series line of mainframes. The new systems rely upon the company's Single Chip A-Series Mainframe Processor, or Scamp.

Two of the four new A6 S models, the dual-processor A6 KS and the four-processor A6 NS, are multiprocessor systems that can be software-partitioned into two independent systems, according to the company. The dual I/O systems are implemented in separate backplanes, and independent power and cooling supplies are available for each processor and cabinet.

Prices for the A-Series S models range from \$150,000 to \$500,000. The partitioning feature, which comes standard on the A6 NS, is available as an option on the A6 KS for \$25,000. The package includes a Unisys PW2 500 PC configured as an operator's console and maintenance processor; plus the software that partitions the system on demand.

Reader Service 33

Wang offers new Tempest VS systems for commercial prices

Wang Laboratories has introduced two Tempest systems, the VS 5660-XT and the VS 8000-T series, with entry-level systems priced at \$55,000 and \$250,000, respectively. The company also says it now offers its Tempest VS computer systems (Tempest VS 6-T, VS 75-T, and VS 7000-T series) at a price equivalent to its commercial VS systems.

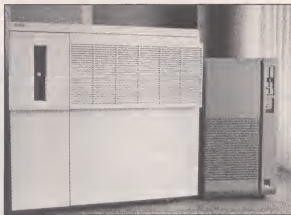
The VS 5660-XT and VS 8000-T systems are software and peripheral compatible with all VS systems and can be configured as stand-alone processors or as end-node systems in distributed networks. They functionally resemble their non-Tempest counterparts, according to Wang, and require no special housing. They are Tempest certified and comply with NACSIM 5100A specifications.

The VS 5660-T is a midrange system that supports up to 90 users and 96 peripheral devices. It reportedly incorpo-

rates a mainframe instruction set into a scalable CMOS chip capable of operating at 33.3 MHz. It supports 802.3 and other communications protocols.

Other features include a single-board CPU, high-performance I/O coprocessor subsystems, and a high-speed system bus. Main memory comes in increments of 1, 2, 4, 8, or 16 Mbytes. Available in June, it will range in price from \$55,000 to \$100,000.

The VS 8000-T sits at the high end of Wang's midrange family. It supports up to 253 users and 255 peripheral devices. The system's single CPU uses a pipelined architecture and CMOS. It maintains compatibility with the VS 7000-T. Prices range from \$250,000 to \$500,000.



Wang offers its new Tempest VS 5660-XT (right) and VS 8000-T (left) series at prices equivalent to its commercial VS systems.

Reader Service 34

Computer and fax join to answer queries for database information

Spectrafax offers an information retrieval and delivery system called Special Request. The system combines synthesized voice communication and facsimile transmission to respond to requests for information sent by customers via fax. It is based on the Spectrafax Personal Link and Voice Messaging products.

Authorized callers can use the touch-tone telephone on a standard Group III fax machine to request and retrieve copies of data from a database stored on a mainframe or PC, or accessed across local area networks. The system responds

to calls using an electronic voice synthesizer. The caller can select one or more documents from the menu of available documents by pressing the appropriate numbers on the fax phone. Pressing the start button on the fax machine causes Special Request to deliver the information requested, whether text or graphics, to the caller's fax machine.

Special Request supports ASCII, PCX, and DCX file formats. It includes scanner, printer, and video drivers, as well as the Special Edition Sfax Graphics Editor. It handles up to 999 data fields contain-

ing up to 999 pages of information each. Certain boards in the system can also handle normal outbound fax traffic.

Host companies can custom configure Special Request to provide service over four to thousands of telephone lines. The tailor-made system costs about \$1,000 per line, with typical configurations using four lines or more.

Demonstrations of Special Request are available by using the handset on your fax machine to call (813) 643-7600.

Reader Service 35

PCs handle digital image processing and enhancement with V

V from Digital Optics provides digital image processing and enhancement on IBM PC AT computers and compatibles. It supports byte, integer, real, and complex images of any dimensions. According to the company, it can interpret complex images in both rectangular and polar forms and supports the conversion of images between types. Users can define an unlimited number of windows on regions within the frame buffer. V commands operate on windows, referred to by name.

An interactive cursor moves across the image and returns regional intensities and location information in several coordinate systems. Users can overlay an unlimited number of erasable symbols, or flags, on image data.

For image contrast and intensity manipulation, V provides intensity range remapping, logarithmic transformation, dual histogram calculation and display, calculation of image statistics, histogram equalization and matching, unimodal and bimodal Gaussian synthetic histogram generation, line profile calculation, intensity interpolation for line profiles, profile plotting, interactive thresholding, and histogram analysis.

Also available are forward and reverse two-dimensional fast Fourier transform, algebraic operations, color, drawing functions, an ASCII character set for text, and geometric operations. Windows and image files in V can carry calibration information describing a user-defined

world-space coordinate system. Spatial convolution is provided with default kernels and a user-definable kernel.

V can operate unsupervised using a command procedure assembled by the user. On-line help is context-sensitive.

V requires an image-processing board, or frame grabber; a high-resolution monitor; a video camera (except in cases where the user has an independent source of digitized images); and a PC AT computer running MS-DOS.

V costs \$1,500 for an individual system. Site licenses are \$3,500 for three complete systems or \$7,500 for 10 complete systems.

Reader Service 36

OOP-based Kappa aids software development on PCs

Intellicorp says that its C-based Kappa software development environment for personal computers combines object-oriented programming, rule-based reasoning, active graphics, and intelligent links to other applications, spreadsheets, and databases.

Application builders reportedly use objects to represent company information directly in software. Rules and other language constructs allow them to capture business-specific decisions and policies. Active graphics, according to the company, permit easy use and incorpora-

tion of dynamic graphics in applications.

The first release of Kappa runs on IBM PC AT or compatible computers with 640 Kbytes of RAM, DOS, and Windows 2.1. The US list price is \$3,500; runtime versions are \$450. The company supports Kappa by telephone hotline and offers a variety of extended maintenance options.

Kappa also gives programmers access to other applications written in Fortran, Pascal, and C.

Reader Service 37

Project supercomputers expand Iris Power Series

Silicon Graphics has extended the Iris Power Series with the 4D/300 line of project supercomputers. According to the company, the new line includes five major enhancements over the existing 4D/200 line: 33-MHz RISC processors; IPI2X disk subsystems; the Power Channel I/O processor; enhancements to the Iris operating system, including React (Real-time Access Technology) real-time extensions to Irix, the company's version of Unix; and high-density 4-Mbit DRAM memory chips.

With the 4D/300 line, Iris Power Series systems now come with up to eight

33-MHz RISC processors. 4D/200 owners can upgrade with 33-MHz CPU boards. 4D/200 owners can also add the Power Channel option through a field board swap.

IPI2X offers 6 Mbytes/s transfer rates. Server configurations of the 4D/300 line range in price from \$74,900 for the two-processor 4D/320S to \$202,500 for the eight-processor 4D/380S. Graphics configurations start at \$114,900 for the 4D/320GTX and \$134,900 for the 4D/320VGX.

Reader Service 38

Brainmaker goes professional

California Scientific Software has announced Brainmaker Professional, based on its Brainmaker program. Brainmaker Professional provides a system for designing, building, training, testing, and running neural networks. This network generation and data manipulation program displays data in spreadsheet format, permits complex arithmetic operations on data, and builds network description and training files automatically.

Brainmaker Professional also features built-in graphics, Fourier analysis capabilities, batch mode running and training, runtime systems and license with distribution rights, and chaotic systems support.

Version 1.0 consists of three parts: *Introduction to Neural Networks*, *Brainmaker Professional Users Guide and Reference Manual*, and *Brainmaker Professional software with Netmaker Professional*. It requires an IBM PC, XT, AT, PS/2, or compatible with 512 Kbytes of memory, monochrome or color display, hard disk drive, and DOS 3.0 or higher. A mouse, 8087 math chip, Desqview, and spreadsheets and graphics packages are optional.

Brainmaker Professional v.1.0 with Netmaker Professional retails for \$795.

Reader Service 39

Encore 90 family will combine Unix, real-time guest operating systems

Encore Computer claims that its new Encore 90 family "power domain" architecture allows multiple real-time and Unix operating systems to work together in one multiprocessing environment. The company has already integrated its MicroMPX and MicroARTE operating systems.

The first products in the family, members of the Encore 91 series, feature an execution environment based on MicroMPX and a development environment, Umax V, based on AT&T's Unix System V. MicroMPX is based on Encore's real-time executive, MPX-32. It independently controls the user-defined system resources, according to the company. Either operating system can reportedly take domain control of the system's hardware.

Initial Encore 91 machines will come in two- or four-processor versions with 16 Mbytes of memory expandable to 272 Mbytes and an enhanced compatible VME-64 bus architecture. They will support two high-speed SCSI channels, plus

Ethernet and X.25 communications standards. The 91 series will employ a 32-bit architecture configured with Motorola's 88000 family of RISC processors.

Real-time features include real-time clocks and timers, hardware interrupts, high-speed data support, and Encore's deterministic reflective memory system, which reportedly allows integration of multiple (up to eight) Encore 91 systems through shared common memory.

Diagnostics include built-in test and remote dial-in capabilities. Encore 91 machines also have an intelligent reconfiguration facility, which allows them to dynamically reconfigure themselves to avoid using faulty elements.

The Encore 91 series will come as desk-side tower cabinets, 19-inch rack-based cabinets, or in a form for integration by others. The company has scheduled initial shipments for the fourth quarter of 1990. Prices for a two-processor, entry-level system will start at \$55,000.

Reader Service 40



Encore Computer's Encore 91 series, first in the Encore 90 family, begins with the four-processor 9104 (background) and two-processor 9102 (foreground), shown here in computer-room models.

Microsystem Announcements

Company, Model, Function	Comments	R.S. No.
Aeon Systems VME 300 Processor board	A processor board incorporating DEC's rVAX 300 subassembly in a 6U board with local memory of 1, 2, 4, or 8 Mbytes and a VME interface. Comes with external connections for Ethernet and RS-232. Ships in late June. Cost: \$4,995 (1 Mbyte).	120
Ariel PC-56D, DAT-56 Daughter cards	Daughter cards with audio and speech capabilities of the Next computer, for IBM PC compatibles. Use Motorola's 56001 DSP chip. Connect to Ariel's digital microphone. PC-56D handles recording, filtering, etc. DAT-56 is a DSP development system PC board. Cost: \$895 for PC-56D; \$1,995 for DAT-56.	121
AST Research Premium 486/33 PC	A desktop PC based on an ISA design and Cupid-32 architecture. Uses a 33-MHz Intel 80486 CPU and 80387-compatible math coprocessor. Supports a Weitek 4167. Comes standard with 4 Mbytes of memory and a 5.25-inch floppy disk drive. Cost: \$9,995 for Model 5; \$11,495 for Model 115 (110-Mbyte hard drive); \$13,645 for Model 325 (320-Mbyte hard drive).	122
Cache Computers Cache486-33/AT, /EISA CPU boards	CPU boards based on Intel's 80486 chip and the AT (ISA) or EISA bus. The EISA version uses Intel's EISA chipset, including the EISA bus buffer controller. The AT version uses the two-chip OPTI chipset and cache controller. Both have up to 16 Mbytes of on-board memory. Cost: \$4,395 for Cache486-33/AT; \$4,895 for Cache486-33/EISA.	123
C&C Technology Vector 32C/8500 DSP board	A 32-bit floating-point vector processor based on AT&T's 50-MHz DSP32C DSP chip. Comes with no external SRAM, or 32 to 512 Kbytes of SRAM and up to 8 Mbytes of DRAM. Works with IBM PC, XT, AT and compatible computers. Cost: \$4,995 (512 Kbytes of SRAM).	124
Compaq 32-Bit Dual-Speed Token Ring Controller	A 32-bit EISA-based token ring network interface controller operating at 16 or 4 Mbps. Features 128 Kbytes of memory, 9-pin subminiature D and RJ-45 connectors, software-selectable interrupt channels, and I/O base address self-determined by EISA slot address. Cost: \$1,299.	125
Datacom Model 8194-III Transient digitizer	A one-megasample/s per channel transient digitizer. Stores data samples in solid-state memory on each digitizer channel card (up to 240,000 samples per card). Controlled by a user's host computer via the GPIB bus. Cost: \$32,000 for 16-channel or \$115,565 for max 64-channel model (256 Kbytes of memory on each channel).	126
Data General Walkabout/SX Laptop PC	A laptop based on Intel's 16-MHz 80386SX CPU. Features a tiltable backlit LCD VGA screen that displays 32 levels of gray. A base configuration has 1 Mbyte of memory, a floppy disk drive, a 40-Mbyte hard disk drive, DOS 4.01, GW-Basic, and LIM 4.0 expanded memory manager. Cost: \$4,995.	127
Datatrace Extract Software	A software tool for database and spreadsheet developers. Reads ASCII reports and exports data into PC file formats including DBF, WK1, and comma delimited. Provides features for conversion to relational databases. Runs under DOS 3.00 or later. Cost: \$279.	128
Dedicated Computer Systems DCS96 Development environment	A PC-based software and hardware development package for the 8096, 8097, 80C196 family of microprocessors. Combines an assembler or C compiler with a source-level debugger and a single-board computer/emulator. Cost: \$495 for assembly language version; \$95 additional for C software.	129
Dell Computer System 425E PC	A PC based on Intel's 25-MHz 80486 CPU and the 32-bit EISA bus. Comes with 4 Mbytes standard; pairs of 1- or 2-Mbyte SIMMs can be added. Has six EISA slots and two ISA slots. Cost: \$8,299 with 4 Mbytes of memory, a floppy disk drive, an 80-Mbyte hard disk drive, and a Super VGA monitor.	130
Dell Computer System 320LX PC	A desktop PC based on Intel's 20-MHz 80386SX CPU. A base configuration has 1 Mbyte of RAM, a 3.5-inch or 5.25-inch floppy disk drive, a 40-Mbyte hard disk drive, seven open expansion slots, and a VGA monochrome monitor. Cost: starts at \$2,899.	131
Dolphin Scientific Desktop Signal Processor DSP unit	A desktop signal processing computer based on AT&T's DSP32C chip. Configurable with 1-18 DSP32C processors. Has 40 analog channels, four 32-bit digital ports, four serial ports, and four RS-232 ports. Connects to IBM PC AT or Macintosh II hosts. Cost: ranges from \$12,500 to \$59,500.	132

Company, Model, Function	Comments	R.S. No.
Industrial CPU Systems ICSI-386 CPU board	An IBM AT-compatible CPU board based on Intel's 80386 chip. Runs at 20 or 25 MHz. Has up to 8 Mbytes of interleaved EMS LIM 4.0-compatible memory available in SIMM modules, a battery-powered clock/calendar, serial and parallel I/O ports, a floppy disk controller, and an IDE hard disk interface. Uses Quadtel BIOS. Cost: \$2,500 (20 MHz, 1 Mbyte) to \$5,100 (25 MHz, 8 Mbytes).	133
Micro Way Number Smasher-486 Motherboard	A replacement motherboard with a 25-MHz Intel 80486 chip. Runs both 16-bit applications and 32-bit protected mode software. Has sockets for up to 16 Mbytes of 32-bit memory and a socket for an optional Weitek 4167 numeric coprocessor. Uses standard AT bus. Cost: \$3,995 (0 Kbytes).	134
Otec Technologies OPC-486-25C, -25X PCs	PCs based on Intel's 80486 CPU. OPC-486-25C has 64 Kbytes of cache memory; OPC-486-25X has 128 Kbytes of cache memory. Both support up to 24 Mbytes of memory. They include 1:1 interleave MFM or ESDI dual disk controllers and have VGA color monitors. Cost: \$4,850 for 25C; \$6,300 for 25X.	135
Sharp Electronics PC-6220 Notebook PC	A notebook-sized PC 1.4 inches high and weighing four pounds. Based on a 12-MHz 80C286 CPU. Has an 80C287 coprocessor socket, 1 Mbyte of standard RAM expandable to 3 Mbytes, a 2.5-inch hard disk drive, DOS 4.01 and Laplink stored in ROM, and standard serial and parallel interfaces. Available in June. Cost: below \$4,000.	136
Yarc Systems Micro785+ Coprocesor system	A coprocessor system for IBM PS/2 Micro Channel applications. Incorporates Motorola's 68020 CISC chip. Has 1, 2, or 4 Mbytes of on-board memory. Up to four Micro785+ coprocessors operate simultaneously in one PS/2. Cost: starts at \$2,795.	137

IC Announcements

Company, Model, Function	Comments	R.S. No.
Integrated Device Tech. IDT49C465 EDC unit	A 32-bit, flow-through, error detection and correction unit with a two-bus architecture. Operates at 15-ns error-detect and 20-ns error-correct times. Also available in a Mil-Std-883C version. Comes in a 144-pin PGA. Cost (OEM quantities): \$92.40 (20-ns correct-time version).	138
Hitachi America HM658512 Series P-SRAMs	A series of 4-Mbit pseudostatic RAMs. Organized as 512 Kwords $\times$ 8 bits. One transistor and one capacitor per memory cell. Contain self-refresh mode. Comes in 600-mil 32-pin DIPs and 525-mil 32-pin surface-mount SOPs. Cost (1,000s): \$80 (100-ns version, DIPs or flat packs).	139
LSI Logic Sparkit Chip set	A chip set for producing workstations compatible with Sun Microsystems' Sparcstation 1. Consists of seven ICs. Sparkit-25 (sampling in June) operates at 25 MHz and comes in plastic quad flat packs or PGAs. Sparkit-40 (available in the second half of 1990) will run at 33 or 40 MHz and will come in ceramic or plastic PGAs. Cost (1,000s): \$1,327 for Sparkit-25.	140
Motorola 68HC05P8, 68HC705P9 Microcontrollers	Two new members of the 68HC05P "P" series of 8-bit microcontrollers. Sampling in second quarter. 68HC05P8 has 32 bytes of EEPROM, 2 Kbytes of ROM, 112 bytes of RAM, and four-channel 8-bit A/D and comes in 28-pin DIPs or SOICs. 68HC705P9 OTP has 2 Kbytes of OTP EPROM, 128 bytes of RAM, SIOP, and four-channel 8-bit A/D. It comes in 28-pin DIPs, SOICs, or windowed ceramic DIPs. Cost (in quantity): \$4 for P8; \$6-\$8 for 705P9.	141
National Semiconductor Flexcell Gate array family	A gate array family with gate density up to 250,000 gates, gate utilization of 50-80%, and improved RAM density over sea-of-gates devices. Built using a 0.8 micron, L-effective CMOS process. Features an architecture with a programmable polysilicon layer. Prices vary.	142
VLSI Technology VM62832 Military SRAM	A low-power, CMOS, 256-Kbit SRAM organized 32,768 $\times$ 8 bits. Has s and cycle times as fast as 45 ns. TTL-compatible, complies with Mil-Std-883 Level B / sampling. Comes in 600-mil, 28-lead, sidebraced ceramic DIPs and 32-lead ceramic 1/ Cost (100s): for 45-ns version, \$144 (DIPs) or \$150 (LLCCCs); for 55-ns version, \$12 s) or \$130 (LLCCCs).	143

CONFERENCE AT A GLANCE



8:30	Breakfast		Session 1A System Scheduling in Multi-processors I		Session 1B System Scheduling in Multi-processors II		Session 1C Compilers and Searching		Session 1D Machine Scheduling and Parallel Systems		Session 1E Machine Diagnosis and Recovery		Session 1F Machine Control and Trees		Session 1G Generalized Networks and Algorithms		Session 1H System Modeling with microcomputer Networks		Session 1I Parallel Processing Systems and Algorithms		Session 1J Dynamic and Specialized Parallel Systems		Session 1K Tutorial 8	
9:45	Breakfast		Session 2A Task Scheduling II		Session 2B Efficient Memory Access Methods		Session 2C Programs and Language Applications		Session 2D Numerical and Applications		Session 2E Routing and Hypertexts		Session 2F Mapping, Cache, and Performance Visualization		Session 2G Fault-Tolerant and Networks		Session 2H Dynamic and Linear Programming		Session 2I Tutorial 5		Session 2J Tutorial 6		Session 2K Tutorial 7	
10:30	Breakfast		Session 3A Performance Analysis and Evaluation		Session 3B High Performance Computing and Programming (Parallel)		Session 3C Image Processing and Techniques		Session 3D Image Processing and Techniques		Session 3E Generalized Networks and Hypertexts		Session 3F Static Modeling for Non-Numeric Applications (Parallel)		Session 3G Parallel Simulation		Session 3H Massively Parallel Systems and Design		Session 3I Tutorial 5		Session 3J Tutorial 6		Session 3K Tutorial 7	
11:30	Breakfast		Session 4A Performance Analysis and Evaluation		Session 4B High Performance Computing and Programming (Parallel)		Session 4C Image Processing and Techniques		Session 4D Image Processing and Techniques		Session 4E Generalized Networks and Hypertexts		Session 4F Static Modeling for Non-Numeric Applications (Parallel)		Session 4G Parallel Simulation		Session 4H Massively Parallel Systems and Design		Session 4I Tutorial 5		Session 4J Tutorial 6		Session 4K Tutorial 7	
12:00	Breakfast		Session 5A Performance Analysis and Evaluation		Session 5B High Performance Computing and Programming (Parallel)		Session 5C Image Processing and Techniques		Session 5D Image Processing and Techniques		Session 5E Generalized Networks and Hypertexts		Session 5F Static Modeling for Non-Numeric Applications (Parallel)		Session 5G Parallel Simulation		Session 5H Massively Parallel Systems and Design		Session 5I Tutorial 5		Session 5J Tutorial 6		Session 5K Tutorial 7	
13:00	Lunch		Session 6A Performance Analysis and Evaluation		Session 6B High Performance Computing and Programming (Parallel)		Session 6C Image Processing and Techniques		Session 6D Image Processing and Techniques		Session 6E Generalized Networks and Hypertexts		Session 6F Static Modeling for Non-Numeric Applications (Parallel)		Session 6G Parallel Simulation		Session 6H Massively Parallel Systems and Design		Session 6I Tutorial 5		Session 6J Tutorial 6		Session 6K Tutorial 7	
14:00	Lunch		Session 7A Performance Analysis and Evaluation		Session 7B High Performance Computing and Programming (Parallel)		Session 7C Image Processing and Techniques		Session 7D Image Processing and Techniques		Session 7E Generalized Networks and Hypertexts		Session 7F Static Modeling for Non-Numeric Applications (Parallel)		Session 7G Parallel Simulation		Session 7H Massively Parallel Systems and Design		Session 7I Tutorial 5		Session 7J Tutorial 6		Session 7K Tutorial 7	
15:00	Lunch		Session 8A Performance Analysis and Evaluation		Session 8B High Performance Computing and Programming (Parallel)		Session 8C Image Processing and Techniques		Session 8D Image Processing and Techniques		Session 8E Generalized Networks and Hypertexts		Session 8F Static Modeling for Non-Numeric Applications (Parallel)		Session 8G Parallel Simulation		Session 8H Massively Parallel Systems and Design		Session 8I Tutorial 5		Session 8J Tutorial 6		Session 8K Tutorial 7	
16:00	Lunch		Session 9A Performance Analysis and Evaluation		Session 9B High Performance Computing and Programming (Parallel)		Session 9C Image Processing and Techniques		Session 9D Image Processing and Techniques		Session 9E Generalized Networks and Hypertexts		Session 9F Static Modeling for Non-Numeric Applications (Parallel)		Session 9G Parallel Simulation		Session 9H Massively Parallel Systems and Design		Session 9I Tutorial 5		Session 9J Tutorial 6		Session 9K Tutorial 7	
17:00	Lunch		Session 10A Performance Analysis and Evaluation		Session 10B High Performance Computing and Programming (Parallel)		Session 10C Image Processing and Techniques		Session 10D Image Processing and Techniques		Session 10E Generalized Networks and Hypertexts		Session 10F Static Modeling for Non-Numeric Applications (Parallel)		Session 10G Parallel Simulation		Session 10H Massively Parallel Systems and Design		Session 10I Tutorial 5		Session 10J Tutorial 6		Session 10K Tutorial 7	
18:00	Lunch		Session 11A Performance Analysis and Evaluation		Session 11B High Performance Computing and Programming (Parallel)		Session 11C Image Processing and Techniques		Session 11D Image Processing and Techniques		Session 11E Generalized Networks and Hypertexts		Session 11F Static Modeling for Non-Numeric Applications (Parallel)		Session 11G Parallel Simulation		Session 11H Massively Parallel Systems and Design		Session 11I Tutorial 5		Session 11J Tutorial 6		Session 11K Tutorial 7	

19th INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

August 13-17, 1990

Location: Pheasant Run Hotel
St. Charles (West of Chicago), IL 60174

CONFERENCE/TUTORIAL REGISTRATION FORM

REGISTRATION FEE: Make your check or money order payable to **INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING** for the TOTAL amount indicated below and mail it to: Pheasant Run, P.O. Box 84, St. Charles, Illinois 60174. Attn: Ms. Ginny Stewart.

Please check appropriate boxes below and enter the total:

☐ Conference	\$210
Pre-Conference Tutorial	
☐ Tutorial 1: Parallel Computing	\$210
☐ Tutorial 2: Artificial Neural Networks & Para. Processing	\$210
☐ Tutorial 3: RISC Multiprocessors & Applications	\$210
☐ Tutorial 4: Tools for Developers of Dist. Software Sys.	\$210
Post-Conference Tutorials	
☐ Tutorial 5: System Integration with Interconnection Ntwks.	\$210
☐ Tutorial 6: Parallel Processing Algorithms and Systems	\$210
☐ Tutorial 7: Dynamic & Speculative Parallelism	\$210
☐ Tutorial 8: Parallel Programming: Compilers & Oper. Sys.	\$210

TOTAL \$ _____

Note:

REFUND POLICY: Cancellation of registration must be in writing to: T. Feng, E.E. East Bldg., The Pennsylvania State University, University Park, PA 16802, received by July 25, 1990. No oral/phone cancellations will be accepted.

We do not accept company/government vouchers or credit cards. Foreign checks must be withdrawable from a U.S. bank.

We reserve the right to cancel a tutorial due to insufficient participation or other unforeseeable problems.

(please print or type)

Mr. Mrs.
Dr. Ms.
Prof./Miss

(First) (Middle) (Last)

Affiliation _____

Address _____

City, State, Zip _____

Business Phone _____ Home Phone _____

If you wish to be on our ICPP mailing list, please write to: T. Feng, E.E. East Bldg., The Pennsylvania State University, University Park, PA 16802

ROOM RESERVATION FORM*

ROOM DEPOSIT: One night deposit payable to PHEASANT RUN, P.O. Box 84, St. Charles, Illinois 60174, Attn: Ms. Ginny Stewart, for guaranteed reservation.

Please fill out the form below:

Arrival Date _____ Departure Date _____

Number of _____ Adults, _____ Children, _____ Rooms

The room rates are:

Standard Room	\$58/single	\$68/double
Deluxe/Tower Room	\$78/single	\$88/double
Blivell Suite (for family, or 3-4 persons)		\$125/suite

Assignment of rooms will be made according to the date of receipt of the room deposit. REGISTER EARLY TO ASSURE ACCOMMODATIONS. Check in time is 4 P.M. The Pheasant Run will try to accommodate early arrivals.

*Room reservation without Conference/Tutorial registration will not be accepted.

IMPORTANT NOTICE

We must receive two checks (one payable to the INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING for conference/tutorial registration, the other for a one-night deposit payable to PHEASANT RUN for room reservation), in order to guarantee both registration and room reservation.

BOTH THE CONFERENCE/TUTORIAL REGISTRATION FORM AND THE ROOM RESERVATION FORM MUST BE RECEIVED BY July 25, 1990.

Send BOTH checks to: Pheasant Run, P.O. Box 84,
St. Charles, Illinois 60174
Attn: Ms. Ginny Stewart

CHECKS MUST BE RECEIVED WITH REGISTRATION AND RESERVATION FORMS

Registration fees must be paid by check or money order, we regret that we do not accept company/government vouchers or credit cards.

Foreign checks must be in U.S. dollar DRAWN ON A U.S. BANK.

Neither phone registration nor registrations without payment will be accepted.

Cancellation of registration must be in writing to: T. Feng, E.E. East Bldg., The Pennsylvania State University, University Park, PA 16802, received by July 25, 1990. No oral/phone cancellations will be accepted.

CONFERENCE DETAILS

LOCATION

Pheasant Run is located in DuPage County, Illinois and is about 25 miles from O'Hare International Airport and 45-minutes from Chicago's loop. The 500-room resort hotel has various sports and health facilities.

TRANSPORTATION

Van Transport from Chicago's O'Hare Airport to the hotel is available through the hotel's Transportation Department at a cost of \$12 per person. A special pickup point is designated for the van: Lower level of the Savan Continnants Restaurant, between Terminals Two (2) and Three (3) at the second curb. For United passengers another special pickup point is located at Door 1G of the United Terminal (which is Terminal 1). All of the vans are brown and tan with a Pheasant Run logo on the side. Advance notice is required. Please call at least three (3) days prior to arrival. If you have any questions or need assistance, call the Transportation Department at (708) 584-6300 - extension 7704. The rates for transportation between the Midway Airport and Pheasant Run are: \$30 for one, \$15 each for two and \$10 each for three or more persons each way.

Additional Copies

The three-volume sets of the ICPP Proceedings may be ordered from the Penn State Press, Suite C, Barbara Bldg., 820 N. University Drive, The Pennsylvania State University, University Park, PA 16802 or by calling (814) 865-1327 or FAX (814) 865-1408. The price is \$125 per set.

Exhibits

Vendor's sessions in parallel with technical sessions are planned for the exhibitors. Interested exhibitors should contact Roger E. Anderson, P.O. Box 808, L-306, Lawrence Livermore National Lab., Livermore, CA 94550, Phone: (415) 422-8572

REGISTRATION

Pre-registration is mandatory because the limited space is being reserved for Conference participants only. THE REGISTRATION DEADLINE is July 25, 1990. Register early to assure your accommodation.

YOU MUST SEND TWO CHECKS:

One for registration for the amount of the conference and/or tutorial, payable to "International Conference on Parallel Processing." See the registration form. One for a guaranteed room reservation. In the amount of one night deposit per room, payable to "Pheasant Run". Please mail both checks to PHEASANT RUN, P.O. Box 84, St. Charles, Illinois 60174, Attn: Ms. Ginny Stewart Phone (708) 584-6300.

Both the registration fee and the deposit for guaranteed reservations will be refunded only if a written cancellation is received on or before July 25, 1990. (See Important Notice above)

FURTHER INFORMATION

For additional information on the technical contents of the Conference, please contact: Dr. Benjamin W. Wah, (217) 333-3516, Coordinated Science Laboratory, University of Illinois, 1101 West Springfield Avenue, Urbana, IL 61801-3082

For information or assistance regarding accommodations or registration, please contact: Ms. Ginny Stewart, Pheasant Run, P.O. Box 84, St. Charles, Illinois 60174. Phone (708) 584-6300.

Sponsored by The Pennsylvania State University

CONFERENCES

Editor: Edmund L. Gallizzi, Computer Science Dept., Eckerd College, St. Petersburg, FL 33733; (813) 864-8272; Compmail+e.gallizzi

Advancing technology stirs up the industry

Ware Myers, Contributing Editor

Technology continues its headlong rush into the future, and the IEEE Computer Society's Compcorn series continues its tradition of trying to foresee where technology is going and what its effects are going to be. The consensus at Compcorn Spring 90 was that, by and large, technology is going to continue to cram more transistors — and, hence, more capability — on each chip. Price per transistor will continue to descend, making more applications feasible. It does not appear that there will be a break in this formula during the 1990s.

The increase in capacity will be so enormous, however, that designers' and marketers' imaginations will be stretched tight trying to find applications that can utilize 20 million logic transistors, for example, on one chip. In contrast, there seems to be agreement that more memory can always be utilized.

Now, if only all this data that these powerful chips will be able to handle could be transmitted from place to place, applications would multiply. Already the telephone network has converted some 2 percent of its wiring to optical fiber, and this medium offers essentially unlimited bandwidth. In the 1990s, optical fiber will become economic, and its use will expand exponentially. The applications this bandwidth will make possible promise to absorb a lot of silicon capacity.

Now, get the word directly from two authorities who spoke at the February 27 opening session of Compcorn Spring 90.

When silicon is "free." Instead of being governed by designing what can be built, as the first era of semiconductor technology was, the industry will be governed in its second era by building what can be designed, Andrew S. Rappaport, president of Technology Research Group, argued in an invited lecture. This change in paradigm results from the continuing exponential development of silicon technology, leading to the insight that "silicon will be free."

The first era, which ended around 1985, could be summed up in the phrase, "Silicon is the world's most expensive

real estate," Rappaport said. Designers and manufacturers were under compulsion to preserve this extremely expensive resource. Fabricators tried to pack circuits close together. They tried to maximize yield in order to lose as little of this precious resource as possible. Thus, to a very large extent, what was designed was governed by what could be built.

Rappaport illustrated what he means by "free" silicon by comparing a state-of-the-art device of 1990 with the likely state of the art in 1999. For today's device, he assumed a 1-micron process, 8-inch wafers, 80 percent yield, and 1-cm² devices. Running through an order-of-magnitude calculation, he arrived at the following numbers.

Theoretical memory density:
9 Mb/in²
300 Mb (or 37.5 MB)/wafer
Logic density:
4 million transistors/in²
120 million transistors/wafer

For 1999 he assumed 0.3-micron technology, 10-inch wafers, and the same yield, resulting in

Theoretical memory density:
150 Mb/in²
7,520 Mb (or 940 MB)/wafer
Logic density:
70 million transistors/in²
3,500 million transistors/wafer

At present, a logic transistor costs about 833 microcents, he said. A bit of memory costs 330 microcents. Projecting to 1999, a logic transistor will cost 29 microcents; a DRAM bit, 14 microcents. Carrying the calculations out to 2010, he crossed the one-microcent line.

To build 1 MB of DRAM in 1989, the theoretical die cost was about \$27, he went on. By 1999, that cost will fall to \$1.10. Similarly, a million-transistor microprocessor chip, now \$8.33, will drop to 29 cents.

At the system level, a typical 15-MIPS Unix workstation with 4 MB of memory now incorporates chips with a die cost of

\$460, or a cost to the system manufacturer of \$1,460, leading to a workstation price of about \$14,000. By 1999, the same calculation yields a price in the range of \$500.

"That is what I mean by free," Rappaport said. "No matter what your assumptions are, if you do the same kind of analysis, you will find the same kind of answers."

At these kinds of costs, a host of products, now too expensive for the markets they would serve, become practical.

Now, before everybody writes in questioning these figures, Rappaport emphasized repeatedly that he was just doing order-of-magnitude calculations based on assumptions that he believes to be possible but that might not entirely come to pass. Future prices may turn out to be several times those in his examples. Nevertheless, his basic point is clear: transistors are going to be very cheap, essentially free.

If that is true, design is going to be "incredibly expensive," he continued. The industry already knows that million-transistor logic chips are very hard to design. By the year 2000, 20-million-transistor chips will be fairly common — and terribly expensive to design, he said.

Trying to get a handle on this cost, he originated a cost-of-design index: dollar-months per gate. Between 1975 and 1990, this figure of merit declined by three orders of magnitude. Projecting it ahead 10 more years, the decline was only one order of magnitude.

He attributed the past decline to the impact of design automation tools. But these tools really have not automated design, he pointed out, they have automated tooling. The tools are very efficient in going from logic diagrams to layouts, but not in analyzing requirements or creating concepts. More efficient ways of doing these upstream tasks are still needed.

"Let's not start with architecture or a logic diagram," he pleaded. "Let's start with an assertion about what we are trying to solve — with requirements capture, concept formulation, the application-to-silicon and application-to-software

transformations. We are never going to get there if we focus on discrete devices or discrete lines of code. We need a breakthrough that will enable us to harness this free silicon."

These breakthroughs are likely to be based on several paradigms for the 1990s, he suggested. The first, patterned after the "Greed is good" motto that characterized the financial community in the 1980s, is "Waste is good." If transistors are free, it may be more efficient to trade transistors for design time or tooling time than to economize their use. It may be more efficient to pull a standard architecture off the shelf, even though it is not precisely tuned to the application, than to spend time developing a better-suited architecture. It may be more efficient to accept standards rather than design more specifically to the application. It may be more efficient to generalize manufacturing and test than to save transistors.

Rappaport's second paradigm is "More expensive is better." It may be a good idea to build costs back up again, at least part way. For example, if the industry has to manufacture in low volume, its costs may turn out to be higher.

In this connection, to keep up with the theoretical growth in semiconductor capacity, "every man, woman, and child will have to double his consumption of transistors every year for the next 10 years," Rappaport figured. "If we assume doubling only every two years — 40 percent compounded annually — by 1999 the industry will need to produce only 4 percent of the current logic wafer starts. This is the sign of a glut."

"I think we are going to see the return of the low-volume facility," he offered. "Responsiveness to the needs of the user is going to be very important."

"I think we are going to see the industry distributed into separate process development, manufacturing, and application companies," he continued. "The real profit is going to come, not from manufacturing semiconductors, but from getting them into the hands of customers, making them useful."

"The company that depends for its



The telecommunications industry will largely convert to optical fiber during this decade, according to Edward Davis, president and chief executive officer of Raynet. This almost unlimited bandwidth will lead to new applications. Meeting this demand will keep the semiconductor fabricators busy.

added value just on the transistors is a dying breed," he said. "A large part of the value is going to be contributed by the software, in terms of microcode in the processors or application code that better exploits the chips."

"Finally, I think the next major breakthrough is going to be the addressing of designs prior to the hardware-software partition. The breakthrough we need is in application-level design or system-level design."

Surging bandwidth to absorb silicon growth. Andy Rappaport this year and Gordon Moore last year (*Computer*, May 1989, pp. 104-107) outlined the exponential growth in semiconductor capacity. Both expressed concern that this capacity might in a few more years outrun the applications that can be found for it. This gap is exemplified by Moore's query: Have you bought your share — four million transistors — yet this year?

Never fear, said Edward M. Davis, president and CEO of Raynet, in the second invited lecture. "I am about to describe the industry that is going to keep the silicon factories very, very busy."

Telecommunications has made extensive use of semiconductors and computer technology all along, but in its basic ap-



Later in this decade, producing logic chips will require facilities that are responsive even at the sacrifice of throughput, said Andy Rappaport of Technology Research Group. "Producing logic in a DRAM fab will be no more appropriate than cooking gourmet meals in a McDonald's franchise."

plication, voice lines, it has resolutely stuck to low fidelity — 3 KHz, Davis noted. The performance of the telephone network, of course, is a function not only of the intelligence of the switches, based on computer technology, but of the transmission capacity of the lines. Until recently, that capacity was limited by the fact that the lines were largely twisted copper pairs. "This bandwidth constraint has prevented the telephone network from carrying many services other than voice," Davis said.

"The next few years will mark the beginning of a bandwidth breakthrough by replacing twisted copper pairs with optical fiber," he continued. "It will be possible to deliver virtually unlimited bandwidth. With this bandwidth will come a range of new services, and the world will truly become a global village."

The telephone industry has standardized on single-mode optical fiber with a core diameter of about 10 microns operating on a laser wave length of 1,300 nanometers, Davis said. Its transmission capacity is theoretically 30 terabit-kilometers per second. This value means that a single fiber could carry a signal of 30 gigabits per second for a thousand kilometers. "That capacity is sufficient to carry all the video signals being propagated on

earth today on one fiber," according to Davis.

"Now, if the 30 Tb-km/s at one wave length is not impressive enough, a single fiber can carry many wave lengths simultaneously," Davis went on. He cited a laboratory experiment at Bellcore (Bell Communications Research) that demonstrated 18 wave lengths on one fiber. He expects research to make it possible in time to put hundreds of wave lengths on a fiber.

Moreover, signal loss in fiber has been greatly reduced in the past 20 years. It became competitive with coaxial cable at 20 db/km in 1970. "Today typical fibers are very close to the theoretical limit, about 0.4 db/km at 1,300-nm wave length," Davis said. That means that the transmission loss in fiber is so low that repeaters can be placed tens of kilometers apart, as opposed to the twisted-pair average of three repeaters per kilometer.

Today, the limiting factor is not fiber. It is laser technology. "Systems in commercial use have a maximum performance of 100 Gb-km/s," Davis said. "This performance is less than one percent of the theoretical possibility of the fiber it-

self. It has increased dramatically since fiber was introduced in the telephone networks about 10 years ago. Those first links had a capacity 1/1,000 of today's links." This performance increase is almost identical to that of other semiconductor-based technology.

At first, fiber was very expensive and not very efficient, Davis said. Only the high volume concentrated on long-distance lines could afford it, but this wiring amounts to only about 0.5 percent of the total in the telephone network. Another 5.5 percent is in the wiring between central offices. The first installation, by AT&T, was between Boston and New York in 1980. By 1986, the first coast-to-coast fiber line had been completed.

By 1989, 3.5 million kilometers of fiber had been installed, Davis said. Even so, this large figure is only 2 percent of the 2 billion kilometers of telephone wires in the United States. Still, about half of all new feeder lines are being installed with fiber. Feeder wiring, running from a central office to remote terminals near a group of subscribers, accounts for about 45 percent of telephone wiring.

"Now, we face the last frontier: fiber in

the local loop," he said. Local loops represent 49 percent of telephone wiring. "With the exception of some limited engineering field trials, there is simply no fiber at all in the local distribution loops."

Local use comes down to a matter of marketing strategy, possible applications, and price. "The initial telephone industry strategy was to find potential sources of additional local revenue that would justify a higher cost [estimated then to be about \$60 per month]," Davis said. The number interested at this price proved to be small.

However, the local loop market is very big. Even now, rebuilding the old copper plant runs to about \$10 billion a year in the US alone. As the demand for local fiber systems develops, their cost may be expected to drop along a learning curve. Much of the system cost, of course, is in computer-like elements, and their cost will continue to decline. If the industry spends from \$20 to \$60 billion a year to replace the current copper system, forecasters estimate a 15- to 25-year replacement period. At any rate, it is a big enough market to absorb much of the fabrication capacity of Silicon Valley, Davis said.

Catching the careful criminal

The all-too-usual reaction when a computer center detects an intruder, Cliff Stoll said at Compton Spring 90, is to disable his access, close the security hole, and notify the users, but otherwise keep the happening close to the vest. This is the wrong reaction, Stoll told an invited-lecture audience. "What we did at Lawrence Livermore National Laboratory was keep the hole open and let the guy in, but monitor everything he did."

Stoll, an astronomy student by background, works in a computer center at LLNL. He is the author of a recent book, *The Cuckoo's Egg*, detailing his experience in tracking criminal hackers. He also reported his experience in *Communications of the ACM* in May 1988.

The possibility of an intruder first came up in August 1986 when a discrepancy in the accounts was noticed. The laboratory decided to monitor the intruder's activity, try to find out what other centers he had penetrated and what damage he had done. His every keystroke was recorded and printed out.

In Stoll's view, when you don't understand something, it is time to do what he calls research. That means apply physical principles, keep a notebook, do your homework, find new methodologies, compile statistics, and trust only proven conclusions. All along, "don't give up." Eventually publish your results, like other researchers. In his case, he had to do his research with no formal budget, zero expertise (to start with), primitive tools, cheap methods, and little help.

Stoll did contact the appropriate law enforcement agencies, but at first they were of little help. With a criminal hacker

they, too, were in novel territory. Over a period of months, Stoll watched this guy invade some 450 computers on the Minnet. He saw him steal long-distance service, identifications, and passwords. The invader exported files. He seemed particularly interested in military files. And through it all, he carefully covered his tracks.

During Stoll's tracking effort, it was necessary to protect the sites the hacker was invading. Stoll asked them to back up their files every day so that little would be lost if the invader took to erasing files. But he did not; he tried to avoid notice. What the invader was doing was monitored in real time so that he could be disconnected if he seemed likely to do any real damage. The few times he was interrupted, the disconnection was made to look like a noisy line.

Eventually, Stoll suspected that the hacker was operating out of Germany. But the hacker's chain from his own computer to LLNL extended through half a dozen intermediate sites. It took time to trace all those telephone links backwards to the invader's home base, and he was cautious. He did not stay on the line long enough for tracing to be completed. Eventually, Stoll created a splendid, lengthy, but phony file he called SDinet. That lured the hacker into downloading it, and that took long enough to trace the call.

The wily hacker turned out to be 26-year-old Marcus Hess of Hanover, West Germany. Just a week and a half before Compton Spring 90, the trial was concluded. The very clever — and cautious — invader received a two-year sentence.

— Ware Myers

"My belief is that within the next two years the cost to install a fiber system will drop to that of copper," he predicted. "Fiber will first be installed to deliver POTS (plain old telephone service) and/or digital services. By the second half of this decade, I believe that very little copper will be reinstalled anywhere in the telephone network. Once fiber has been installed, its inexpensive and unused bandwidth will permit services never before delivered or previously considered."

"One thing is certain," he added. "It is only a question of time until every home has greater bandwidth delivered to it than the largest corporations today."

That leaves to the marketplace — and the imagination — the uses to be made of this bandwidth. Perhaps a number of services that have been talked about for years will become practical: videotex, home security, energy management, meter reading, custom calling, voice store and forward, videophone.

Perhaps cable television will turn from

coaxial cable to the fiber network. Farther out, instead of using the limited radio spectrum to broadcast radio and television to everybody, these services will go out over fiber, reserving the air waves for point-to-point communication, he indicated. That might, in turn, solve one of the problems of high-definition television — where to get the bandwidth to carry the additional pixels. Still farther out, Davis envisaged custom television, where you could order, for example, the type of news you want when you want it.

With greater bandwidth, working at home may blossom further. Activities that require the transmission of large volumes of real-time data would become feasible, Davis pointed out.

Interactive services, such as electronic applications for voter registration and passports, may grow up. Davis observed that a host of existing services provided over the telephone require a human translator between the consumer and some computer database. With more computer

and transmission capacity, much of this translation function could be performed by software.

"All the applications spawned by this upcoming bandwidth breakthrough will use computer technologies, both hardware and software," Davis said in closing. "You will both enable — and benefit from — this next step in communications functions. The bandwidth will be out there. Your challenge is to fill it with the applications that will enrich all our lives."

Comcon Spring 90, held February 27-March 1 in San Francisco, was chaired by Kenichi Miura of Fujitsu America. Roger Anderson of Lawrence Livermore National Laboratory served as program chair. The *Digest of Papers* is available from the Computer Society Press, Los Alamitos, California. For price and order information, call 1-800-CS-Books or, in California, (714) 821-8380; reference order number 2028.

Can women engineers alleviate the shortage?

"There is a shortage of engineers, both the number of college students majoring in engineering and the number that then stay in engineering," Donna K. Potter told the audience at a Comcon Spring 90 panel titled "Tapping 51 percent of the Population." An engineer at Lockheed Missile and Space, Potter is also president of the Santa Clara Valley Section of the Society of Women Engineers.

Interest in the physical sciences by college students dropped from 3 percent to 1.5 percent in the last decade, she said. "Last year, less than 1 percent of college freshmen said they intended to major in mathematics, compared with 4 percent 20 years ago." Then, even if this small percentage of students graduate — and in engineering only 70 percent do — the defection rate to other occupations is greater than in any other field.

Encouraging more girls to study mathematics and science and making them feel comfortable in scientific and engineering programs can help overcome these shortages. "Women are making significant inroads in medicine, law, and business," she said, "yet their interest in engineering has tapered off at an alarming rate since the early eighties."

Working on the nuts and bolts of the effort to interest girls in mathematics is Alice J. Kelly of Santa Clara University, national director of the Women and Mathematics Program. The program is sponsored by the Mathematical Association of America and is funded by IBM, GI Alden, John Hancock, Hewlett-Packard, and recently Northern Telecom, among others, she said. "We provide role models, career counseling, academic advice and with luck, expose [young women] to the beauty and joy of mathematics."

Moreover, there are actions we can all take in our own homes, she pointed out. Both mothers and fathers should help both boys and girls with their math and science home-

work — without discrimination in either direction. Ensure that female children are encouraged the same as males to play with technical toys such as Erector sets, Lego blocks, electronic kits, and chemistry sets. Point out role models of both sexes in technical occupations. "Involve your children in the use of mathematics in daily life," she suggested. "Have them help compute gas mileage, time to get to your destination when traveling, discounts on purchases, and so on."

"When was the last time you felt different?" Rebecca A. Failor, a nuclear chemist at Lawrence Livermore National Laboratory, asked the mostly male audience. "Now, try to imagine how it feels to be the only technical woman on your staff."

Once a young woman becomes interested in math and science, once she acquires an education in a technical field, she still faces a psychological hurdle in the workplace itself. Sometimes, it is as simple as the look of surprise on the male faces when "R. Failor" first arrives. Sometimes, it is as tricky as the "girlie" posters that adorn some male offices. "How can you do business with this man when right over your shoulder he is looking at his dream blonde draped over the hood of a Corvette?" she asked.

One area where nature has made it very hard to ignore the differences is child-bearing. Failor recommends: Openly discuss with pregnant women ways to handle their jobs during their pregnancy and maternity absences.

"Treating women as equals, as workers who are basically the same as their peers, gives women the freedom to be their best," Failor said. "Your company will find it easy to keep the women you have and 'find' women to promote into the managerial ranks when the women you hire are made to feel that they truly belong as valued members of your technical staff."

— Ware Myers

Kung enthalls symposium audience with Warp tale

Nancy Hays, Associate Editor

Parallel processing is not a mature field — it's more on the kindergarten level. So said H.T. Kung of Carnegie Mellon University in his April 6 keynote speech to an attentive audience at the Fourth Parallel Processing Symposium, held April 4-6 at the University Center on the California State University, Fullerton, campus.

Kung explained that microprocessors don't communicate well, so don't work well for parallel processing. The exception — transputers — are too small. Those who criticize parallel systems for their inefficiency in handling sequential work miss the point that the proper environment for handling a mix of parallel and sequential processing is a heterogeneous one. Such an environment demands a good network.

Larry Canter of Computer Systems Approach, general chair of the symposium, introduced Kung's talk on Carnegie Mellon's Warp projects.

After sketching Carnegie Mellon's Warp, iWarp, and Nectar projects, Kung detailed the makeup of the Warp and iWarp parallel processing chips. The Warp chip was one of the first to allow systolic computation and to attempt to support compilers, he said. The succeeding iWarp chip has 650,000 transistors

and an architecture more complex than that of Intel's 486 chip. The iWarp chip has both a computation agent and a communication agent, which permits separate control and close cooperation between the two. Close connection with local memory further benefits communication.

According to Kung, the iWarp's logical channels constitute a unique feature that meets the need of systolic processing for long-lived connections by reserving bandwidth. The chip features a single-cycle, wide-instruction architecture. As Kung pointed out, the logical channels also permit reconfiguration of the systems into a fault-tolerant organization.

Coming back to his assertion that parallel processing really needs a good network, Kung dismissed Ethernet and FDDI networks as inadequate. He explained that Carnegie Mellon's Nectar project attempts to increase communication by giving each machine its own connection to the network. Nectar-Net includes communication accelerators and several central hubs connecting a variety of machines. A prototype built in 1989-90 connected four hosts. The next step is a 32-host Nectar-Net.

An upcoming version called Gigabit

Nectar, to be constructed with the aid of an industrial partner, will have 64 hosts. Gigabit Nectar will connect Cray Y-MP supercomputers, iWarp systems, and workstations. Despite his reluctance to publicize schedules, Kung said that the group plans to construct the testbed during 1990-92.

After his talk, Kung was presented a symposium plaque of appreciation entitled the Charles Babbage Award. The second such Babbage award, the plaque acknowledges the recipient scientist's significant contributions to society, such contributions seldom being recognized.

The symposium was cosponsored by the Orange County chapter of the IEEE Computer Society and Cal State Fullerton. Nadar Bagherzadeh of the University of California at Irvine and Sandeep Gulati of the Jet Propulsion Laboratory served as technical co-chairs. V.K. Prasanna Kumar of the University of Southern California served as program committee chair.

The proceedings is available from the CS Press, Los Alamitos, California, by calling (800) CS-BOOKS or (714) 821-8380 in California or from OC IEEE Parallel Processing Symposium, 1619 N. Hale Ave., Fullerton, CA 92631.

NCGA 90 keynoter Platt sizes up fast-growing computer graphics field

Michael Haggerty, Assistant Editor, IEEE Computer Graphics & Applications



Computer graphics will solidify its key role in the field, HP's Lewis Platt told an NCGA 90 audience March 19.

The computer graphics market has expanded 10-fold into a \$20-billion industry in the past decade, said Hewlett-Packard's Lewis E. Platt when he keynoted NCGA 90 March 19 in Anaheim, California. Platt spoke at the National Computer Graphics Association's 11th annual conference and exposition featuring 259 exhibitors and 155 conference sessions.

Purportedly the largest yearly event of its kind devoted to a wide range of computer graphics topics, the four-day NCGA 90 attracted nearly 30,000 attendees. This year's theme was "Real Solutions to Real Problems." Alan Christman of the CIM Marketing Group served as conference director.

Platt, who is HP's executive vice president responsible for computer products, began his talk, "Computer Graphics: Where Do We Go Now That We've Arrived?" by illustrating the strength of the computer graphics market. Among the striking statistics he used was the com-

parison of 1979's \$2.1-billion market figure with today's roughly ten-times-larger one.

Platt explained that, because human beings learn more through sight than through any other sense, computer graphics will continue to play an important part in the computer industry. As an example of visual learning, Platt stressed the importance of scientific visualization in helping scientists and lay people understand complex concepts. Visualization is especially helpful in such fields as meteorology and aerospace engineering, said Platt, adding that NASA regularly uses scientific visualization to help the US Congress understand its projects.

Platt listed the four goals facing computer graphics professionals and the graphics industry in the upcoming years: realism in computer-aided design and 3D modeling, speed to reduce design costs and decrease the time it takes to get new products to market, creative use of graph-

ics standards, and the integration of computer graphics with other media such as video and high-definition television.

Throughout the four days of the conference, exhibitors displayed innovations in computer graphics hardware, software, and services. Conference sessions ranged from business presentation systems to aerospace and military applications, from computer-aided design and engineering to computer graphics in the arts.

The conference also featured NCGA's annual awards dinner, a showcase of computer-animated films, and Integrate 90, a demonstration of systems integration across a number of systems, departments, and applications.

'Engineering the Future' to be focus of DAC 90 keynotes

Focusing on new computing technologies and emerging industry standards, Bill Joy of Sun Microsystems will deliver the keynote speech June 25 when the 27th Design Automation Conference convenes at the Orlando (Florida) Orange County Convention Center.

Joy, cofounder and vice president for research and development of Sun Microsystems, will speak on "Engineering the Future." In his discussion of a number of industry-related issues, he will offer his views on the far-reaching innovations anticipated in the computer industry during the 1990s. He will address topics like the distributed applications of tomorrow and other next-generation uses for networked systems.

The week-long DAC 90 is sponsored by the IEEE Computer Society's Design Automation Technical Committee and ACM's Special Interest Group on Design Automation. Richard C. Smith of National Semiconductor is this year's general chair, and Alfred E. Dunlop of AT&T Bell Labs is serving as program chair.

The technical program will feature 125 technical papers, six full-day tutorials, and panel discussions. The tutorials, to be held June 28, will cover the latest developments in electronic design automation.

Registration and tutorial discounts are available to those who sign up on or before May 25. Additional information can be obtained by writing to 27th Design Automation Conference, 7490 Clubhouse Rd., No. 200, Boulder, CO 80301, or calling (303) 530-4333.

The proceedings will be available from the Computer Society Press, Los Alamitos, California, by calling (800) CS-BOOKS or (714) 821-8380 in California.

CALENDAR

 In the accompanying Calendar, the IEEE Computer Society logo identifies the conferences the society is sponsoring and participating in. Other conferences of interest to our readers, as well as their sponsors, are also listed.

For inclusion in Call for Papers or Calendar, submit information at least **six weeks before the month of publication** (i.e., for the July 1990 issue, send information for receipt by May 15, 1990) to Chuck Governale, Calendar Dept., Computer, PO Box 3014, Los Alamitos, CA 90720-1264.

May 1990

43rd SPSE Conf., May 20-25, Rochester, N.Y. Sponsor: SPSE (Society for Imaging Science and Technology). Contact Michael M. Shahin, Xerox Corp., Webster Research Center, 800 Phillips Rd., 0114-38D, Webster, NY 14580, phone (716) 422-2011.

Neural Networks: Concepts, Applications, and Implementations, May 20-26, Dubrovnik, Yugoslavia. Sponsor: European Center for Peace and Development. Contact N. Ostojic, NCPD, Kneza Mihaila 7, Floor 2, 11000 Beograd, Yugoslavia, fax 38 (11) 623-169, e-mail cett026@yubgss21.bitnet.

Pacific Northwest Software Quality Conf. Spring Workshops, May 21, Beaverton, Ore. Sponsor: PNSQC Committee. Contact Terri Moore, Pacific Agenda, PO Box 10142, Portland, OR 97210, phone (503) 223-8633.

Second Int'l Conf. on Computer-Integrated Manufacturing, May 21-23, Troy, N.Y. Cosponsors: Northeast Manufacturing Technology Center, Rensselaer Polytechnic Inst. Contact Sheila Nason, Rensselaer Polytechnic Inst., Troy, NY, 12180-3590, phone (518) 276-6098.

 **Fifth Conf. on Artificial Intelligence for Space Applications, May 22-23, Huntsville, Ala.** Cosponsors: IEEE Computer Society Huntsville Chapter et al. Contact Continuing Education Div., Univ. of Alabama in Huntsville, Tom Bevill Center 285-1, Huntsville, AL 35899, phone (800) 448-4035 or (205) 895-6372.

Ninth Classified Military Space Symp., May 23-24, Washington, DC. Sponsor: American Astronautical Society et al. Contact AAS, 6352 Rolling Mill Pl., Suite 102, Springfield, VA 22152, phone (703) 866-0020, fax (703) 866-3526.

 **First Conf. on Visualization in Bio-medical Computing, May 22-25, Atlanta.** Cosponsors: Nat'l Science Foundation et al. Contact Norberto Ezguerra, Bioengineering

Center, Georgia Tech, Atlanta, GA 30332, phone (404) 894-7026 or 3964.

SIGMetrics 90, May 22-25, Boulder, Colo. Sponsor: ACM. Contact Gary J. Nutt, Univ. of Colorado, Boulder, CO 80301; or Herb Schwetman, MCC, 3500 W. Balcones Center Dr., Austin, TX 78759, phone (512) 338-3428.

 **20th Int'l Symp. on Multiple-Valued Logic, May 23-25, Charlotte, N.C.** Contact George Epstein, Computer Science Dept., Univ. of North Carolina at Charlotte, 214 Kennedy Bldg., Charlotte, NC 28223, phone (704) 547-4566; or Carolyn F. Blatock, Office of Continuing Education, Univ. of North Carolina at Charlotte, Charlotte, NC 28223, phone (704) 547-4861.

1990 American Control Conf., May 23-25, San Diego, Calif. Sponsor: American Automatic Control Council. Contact Daffin Gangsaas, Boeing Advanced Systems, PO Box 3707, MS 33-12, Seattle, WA 98124-2207, phone (206) 393-6852.

ICCI 90, Int'l Conf. on Computing and Information, May 23-26, Niagara Falls, Canada. Sponsor: Natural Sciences and Engineering Research Council of Canada. Contact Waldeemar W. Koczodaj, ICCI 90, Laurentian Univ., CoSc, Sudbury, Ont., Canada P3E 2C6.

 **Mainframe/PC Cooperative Processing, May 24, New York City.** Sponsor: New York Chapter, IEEE Computer Society. Contact Jim Barbera, NYNEX Service Co., 1166 Ave. of the Americas, Rm. 8N1, New York, NY 10036, phone (212) 395-8765.

 **17th Int'l Symp. on Computer Architecture, May 28-31, Seattle.** Cosponsors: ACM. Contact Jean L. Baer or Larry Snyder, Univ. of Washington, Computer Science Dept., FR-35, Seattle, WA 98195, phone (206) 543-1695.

 **Euro ASIC 90, May 28-June 1, Paris.** Contact Gabriele Saucier, Inst. Nat'l Polytechnique de Grenoble/CSI, 46, ave. Felix Viallet, 38931 Grenoble Cedex, France, phone 33 (1) 76-57-46-87.

ICDCS 10, 10th Int'l Conf. on Distributed Computing Systems, May 28-June 1, Paris. Sponsor: INRIA. Contact R. Papescu-Zeletin, GMD-FOKUS, Hardenbergplatz 2, D-1000 Berlin 12. West Germany, phone 49 (30) 25499-206; Jack Stankovic, Computer and Information Science Dept., Univ. of Massachusetts, Amherst, MA 01003, phone (413) 545-0720; or ICDCS 10, IEEE Computer Society, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

CRAI Spring Int'l Seminar on Highly Parallel Processing: Architectures, Programming Tools, and Applications, May 28-June 1, Capri, Napoli, Italy. Sponsor: Consorzio per la Ricerca e le Applicazioni di Informatica. Contact Francesca Scornajenghi, CRAI, Via Tevere 15, 00198 Roma, Italy, phone 39 (6) 852-500, fax 39 (6) 851-727.

11th Conf. of the Canadian Applied Mathematics Society, May 29-June 1, Halifax, N.S., Canada. Cosponsors: CAMS et al. Contact Mary Meidell, Continuing Education Dept., Technical Univ. of Nova Scotia, PO Box 1000, Halifax, NS, Canada B3J 2X4, phone (902) 429-8300, ext. 2420.

June 1990

CBMS 90, Third IEEE Symp. on Computer-Based Medical Systems, June 3-6, Chapel Hill, N.C. Cosponsor: IEEE Engineering in Medicine and Biology Society. Contact James N. Brown, Jr., Research Triangle Inst., 3040 Cornwallis, Research Triangle Park, NC 27709, phone (919) 541-9675.

Spring Comdex, June 3-6, Atlanta. Contact Interface Group, 300 First Ave., Needham, MA 02194, phone (617) 449-6600.

IEEE Infocom 90, Ninth Conf. on Computer Communications, June 3-7, San Francisco. Cosponsor: IEEE Communications Society. Contact Infocom 90, IEEE Computer Society, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

LICS 90, Fifth Symp. on Logic in Computer Science, June 4-7, Philadelphia. Contact Albert Meyer, Computer Science Lab, 545 Technology Square, NE 43-315, Cambridge, MA 02139, phone (617) 253-6024.

Int'l Workshop on Rapid System Prototyping, June 5-7, Triangle Research Park, N.C. Contact Kenneth Anderson, Siemens Corporate Research, 755 College Rd. E., Princeton, NJ 08540, phone (609) 734-6550.

Eurographics Workshop on Object-Oriented Graphics, June 6-8, Konigswinter, West Germany. Sponsors: Eurographics and German Society for Informatics. Contact Marja Hegt, O-O Graphics Workshop, CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands, phone 31 (20) 592-4058.

Sixth ACM Symp. on Computational Geometry, June 6-8, Berkeley, Calif. Contact Raimund Seidel, Computer Science Div., Univ. of California at Berkeley, Berkeley, CA 94720, phone (415) 692-5250, e-mail seidel@ernie.berkeley.edu.

ACL 90, 28th Conf. of the Assoc. for Computational Linguistics, June 6-9, Pittsburgh. Contact Don Walker, Bellcore, MRE 2A379, 445 South St., Box 1910, Morristown, NJ 07960-1910, phone (201) 829-4312.

1990 European Simulation Multiconf., June 10-13, Nuremberg, West Germany. Sponsor: Society for Computer Simulation Int'l. Contact Bernd Schmidt, Univ. Erlangen, Computer Science Dept., D-8520, Erlangen, FRG, phone (49) 9131-857-278; or SCS Int'l, c/o Philippe Gerli, Coupure Links 653, B-9000 Ghent, Belgium, phone (32) 91-23-69-61.

Int'l Workshop on Algorithms and Parallel VLSI Architectures, June 10-16, Pont-A-Mousson, France. Cosponsors: IEEE, Eurasp. Contact Alle-Jan van der Veen, Electrical Engineering Dept., Delft Univ. of Technology, 2628 CD Delft, The Netherlands, phone (31) 1578-1442.

IFIP Workshop on Design and Test of ASICs, June 11-12, Hiroshima, Japan. Cosponsors: Information Processing Society of Japan et al. Contact Kozo Kinoshita, Hiroshima Univ., 1-1-80 Higashisendacho, Nakaku, Hiroshima-shi 730, Japan, phone 81 (87) 249-9150.

Eurographics Workshop on Photosimulation, Realism, and Physics, June 11-13, Rennes, France. Sponsor: INRIA-IRISA. Contact Kadi Bouatouch, IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France, phone (33) 9936-2000, fax (33) 9938-3832, e-mail kadi@irisa.fr.

CTRS 90, Second Int'l Workshop on Conditional and Typed Rewriting Systems, June 11-14, Montreal, Canada. Cosponsors: Centre de Recherche Informatique a Montreal, Concordia Univ. Contact Mitsuhiko Okada, Computer Science Dept., Concordia Univ., 1455 de Maisonneuve Ouest, Montreal, H3G 1M8, Canada, phone (514) 848-3048, e-mail ctrs@concordia.cs.concordia.ca.

19th Mumps Users' Group Meeting, June 11-15, Orlando, Fla. Contact Mumps Users' Group, 4321 Hartwick Rd., Suite 100, College Park, MD 20740, phone (301) 779-6555.

1990 ACM Int'l Conf. on Supercomputing, June 11-15, Amsterdam. Contact E. Gallipoulos, Univ. of Illinois CSRD, 305 Talbot Lab, 104 S. Wright St., Urbana, IL 61801-2932 (for North and South America); John R. Gurd, Computer Science Dept., Univ. of Manchester, Oxford Road, Manchester M13 9PL, UK (Europe and Africa); or Yoichi Muraoka, Electrical Engineering Dept., Waseda Univ., 3-4-1 Okubo, Shinjuku-ku, Tokyo, Japan (Japan and the Far East).

Usenix Summer 1990 Technical Conf., June 11-15, Anaheim. Contact John R. Mashey,

Anaheim Usenix Technical Program, MIPS Computer Systems, 930 Arques Ave., Sunnyvale, CA 94086, phone (408) 991-0253; or Usenix Conf. Office, 22672 Lambert St., Suite 619, El Toro, CA 92630, phone (714) 588-8649.

Eight Int'l Congress on Cybernetics and Systems, June 11-15, New York City. Cosponsors: IEEE, Hunter College. Contact Kathy Jaeger, Hunter College, City Univ. of New York, Computer Science Dept., 695 Park Ave., New York, NY 10021, phone (212) 772-5213.

Computer Security Foundations Workshop III, June 12-14, Franconia, N.H. Contact Tom Haigh, Secure Computing Technology Corp., 2855 Anthony Lane South, Suite 130, St. Anthony, MN 55418, phone (612) 782-7145.

Ninth Int'l Conf. on Analysis and Optimization of Systems, June 12-15, Antibes, France. Sponsor: INRIA. Contact Conf. Secretariat, INRIA, Service des Relations Exterieurs, Domaine de Voluceau-BP 105, 78153 Le Chesnay Cedex, France, phone 33 (1)-39-63-5500.

Tenth Rochester Forth Conf. on Embedded Systems, June 12-16, Rochester, N.Y. Contact Lawrence P. Forsley, Inst. for Applied Forth Research, 70 Elmwood Ave., Rochester, NY 14611, phone (716) 235-0168, fax (716) 328-6426, e-mail L.Forsley on Genie and L.Forsley on Bix and Delphi.

Third Workshop on Methods and Tools for Reuse, June 13-15, Syracuse, N.Y. Contact Bill Frakes, Software Productivity Consortium, SPC Bldg., 2214 Rock Hill Rd., Herndon, VA 22072, phone (703) 742-7108.

IAPR Workshop on Syntactic and Structural Pattern Recognition, June 13-15, Murray Hill, N.J. Sponsor: Int'l Assoc. for Pattern Recognition. Contact Henry S. Baird, AT&T Bell Laboratories, Rm. 2C-557, 600 Mountain Ave., Murray Hill, NJ 07974, phone (201) 582-5744.

10th Int'l Symp. on Protocol Specification, Testing, and Verification, June 13-15, Ottawa, Ont., Canada. Sponsor: IFIP. Contact Luigi Logrippo, Computer Science Dept., Univ. of Ottawa, Ottawa, Ont., Canada K1N 6N5.

IAPR TC7 Workshop on Multisource Data Integration in Remote Sensing, June 14-15, College Park, Md. Cosponsors: Int'l Assoc. for Pattern Recognition, NASA Goddard Space Flight Center. Contact James C. Tilton, MC 636, NASA Goddard Space Flight Center, Greenbelt, MD 20771, phone (301) 286-9510.

Third Int'l Symp. on Image Compression, June 17-20, Rochester, N.Y. Sponsor: Society for Imaging Science and Technology. Contact Charlton Bard, 74 Cornwall Lane, Rochester, NY 14617, phone (716) 342-3174.

ICPR 90, 10th Int'l Conf. on Pattern Recognition, June 17-21, Atlantic City,

N.J. Sponsor: International Assoc. for Pattern Recognition. Contact Herbert Freeman, CAIP Center, 605 Hill, Rutgers Univ., New Brunswick, NJ 08903, phone (201) 932-4208; or, for workshop, Cathy Davison, Computer Science Dept., Michigan State Univ., East Lansing, MI 48824, phone (517) 355-5218.

Third IFIP Workshop on Geometric Modeling, June 17-21, Rensselaerville, N.Y. Contact Mary Johnson, Rensselaer Design Research Center, Rensselaer Polytechnic Inst., Troy, NY 12180-3590, phone (518) 276-6751.

ICNN 90, 1990 Int'l Joint Conf. on Neural Networks, June 17-21, San Diego, Calif. Cosponsors: IEEE, Int'l Neural Network Society. Contact Nomi Feldman, UCNN, 5665 Oberlin Dr., Suite 110, San Diego, CA 92121, phone (619) 453-6222.

EDFT, Seventh European Workshop on Design for Testability, June 18-20, Segovia, Spain. Contact T.W. Williams or C. Lopez Barrio, IBM, PO Box 1900, Dept. AJA/021B, Boulder, CO 80301-9191, phone (303) 924-7692.

IMSC 90, Int'l Mobile Satellite Conf., June 18-20, Ottawa, Canada. Cosponsors: NASA, Canadian Dept. of Communications. Contact IMSC 90 Organizing Committee, c/o D. Hugh M. Reekie, Dept. of Communications, 300 Slater St., Ottawa, Ont., Canada K1A 0C8.

Workshop on Computer-Aided Verification, June 18-20, Princeton, N.J. Contact E.M. Clarke, Computer Science Dept., Carnegie Mellon Univ., Pittsburgh, PA 15213-3890; R.P. Kurshan, AT&T Bell Labs, Rm. 2C-353, Murray Hill, NJ 07974; A. Pnueli, Weizmann Inst., Rehovot, Israel; or J. Sifakis, LIG-IMAG, BP 53X, 38041 Grenoble Cedex, France.

Seventh Int'l Conf. on Testing Computer Software, June 18-21, San Francisco. Contact Genevieve Houston-Ludlam, ISTC 90, c/o Frontier Technologies, 190 Admiral Cochran Dr., Suite 180, Annapolis, MD 21401, phone (301) 266-8244.

SIGPlan 90, Conf. on Programming Language Design and Implementation, June 18-22, White Plains, N.Y. Sponsor: ACM. Contact Mark S. Johnson, Sun Microsystems, 2550 Garcia Ave., MS 12-40, Mountain View, CA 94043-1169, phone (415) 336-7758, e-mail markscotjohnson@eng.sun.com.

Second Int'l Conf. on Software Engineering and Knowledge Engineering, June 21-23, Skokie, Ill. Sponsors: Knowledge Systems Inst. of Pittsburgh, and Inst. for Information Industries, Taiwan. Contact Shi-Kuo Chang, Computer Science Dept., Univ. of Pittsburgh, 322 Alumni Hall, Pittsburgh, PA 15260, phone (412) 624-8490.

DAC 90, 27th ACM/IEEE Design Automation Conf., June 24-29, Orlando, Fla. Contact Pat Pistilli, MP Associates, 7490 Clubhouse Rd., Suite 102, Boulder, CO 80301, phone (303) 530-4333.

NECC 90, 11th Nat'l Educational Computing Conf., June 25-27, Nashville, Tenn. Sponsor: Int'l Council for Computers in Education. Contact John D. McGregor, Computer Studies Dept., Murray State Univ., Murray, KY 42071, phone (502) 762-2614.

Int'l Symp. on Fuzzy Approach to Reasoning and Decision Making, June 25-29, Bechyně, Czechoslovakia. Sponsor: Int'l Fuzzy System Assoc. Contact Vilem Novak, Minin Inst., Czechoslovakia Academy of Sciences, A. Rimana 1768, 70800 Ostrava-Poruba, Czechoslovakia.

Advanced Research Workshop on 3D Imaging in Medicine, June 25-29, Travemünde, West Germany. Sponsor: NATO. Contact Linda Houseman, Computer Science Dept., Univ. of North Carolina, Box 3175, Sitterson Hall, Chapel Hill, NC 27599, phone (919) 962-1758 (for the Americas); or Andreas Pömmert, Inst. für Mathematik und Datenverarbeitung in der Medizin, Univ. Krankenhaus Eppendorf, Martinistrasse 52, 2000 Hamburg 20, Federal Republic of Germany, phone 49 (40) 468-2300 (for Europe, Asia, Australia, and Africa).

EKAU 90, Fourth European Knowledge Acquisition for Knowledge-Based Systems Workshop, June 25-29, Amsterdam. Contact John H. Boose, Advanced Technology Center, Boeing Computer Services 7L-64, PO Box 24346, Seattle, WA 98124, phone (206) 865-3253.

FTCS 20, 20th Int'l Symp. on Fault-Tolerant Computing, June 26-28, Newcastle upon Tyne, England. Cosponsors: Centre for Software Reliability, British Computer Society, IEEE. Contact Neil Speirs, Computing Lab, Univ. of Newcastle upon Tyne, Newcastle upon Tyne, NE1 7RU, UK, phone 44 (91) 232-8511.

Compass 90, Fifth Conf. on Computer Assurance: Systems Integrity, Software Safety, and Process Security, June 26-29, Gaithersburg, Md. Cosponsors: IEEE Aerospace and Electronics Society, IEEE Nat'l Capital Area Council. Contact Dolores Wallace, Nat'l Inst. of Standards and Technology, Gaithersburg, MD 20899, phone (301) 975-3340.

CGI 90, Computer Graphics Int'l 1990, June 26-30, Kent Ridge, Singapore. Sponsors: Computer Graphics Society, Inst. of Systems Science, Singapore. Contact Juzi Motiwalla, CGI 90, ISS, Nat'l Univ. of Singapore, Heng Mui Keng Terr., Kent Ridge, Singapore 0511, phone (65) 772-2075.

1990 ACM Conf. on Lisp and Functional Programming, June 27-29, Nice, France. Contact Gillies Kahn, INRIA Sophia — Antipolis, 2004 Route des Lucioles, 06565 Valbonne Cedex, France, phone (33) 93-65-78-01.

Conf. on Visual Information Assimilation in Man and Machine, June 27-29, Ann Arbor, Mich. Contact Univ. of Michigan Extension Service, Conference and Institutes Dept., 200 Hill St., Ann Arbor, MI 48104-3297, phone (313) 764-5305.

Fifth Rocky Mountain Conf. on Artificial Intelligence, June 28-30, Las Cruces, N.M. Cosponsors: ACM et al. Contact Paul McKeivitt, Computing Research Lab, Dept. 3CRL, Box 30001, New Mexico State Univ., Las Cruces, NM 88003-0001, phone (505) 646-5109, fax (505) 646-6218, Internet paul@nmsu.edu.

July 1990

Roundtable Discussion on Vision-Based Vehicle Guidance, July 2, Tokyo. Sponsor: Committee of IEEE Int'l Workshop on Intelligent Robots and Systems. Contact Ichiro Masaki, Computer Science Dept., GM Research Labs, 30500 Mound Rd., Warren, MI 48090-9055, phone (313) 986-1466.

DPDS 90, Second Int'l Symp. on Databases in Parallel and Distributed Systems, July 2-4, Dublin, Ireland. Cosponsors: ACM et al. Contact Rakesh Agrawal, AT&T Bell Labs, Rm. 3D450, 600 Mountain Ave., Murray Hill, NJ 07974, phone (201) 582-2250; or David Bell, Inst. of Informatics, Univ. of Ulster, Jordanstown, County Antrim, Northern Ireland BT370QB, phone (0232) 365-131.

Second Int'l Conf. on Economics and Artificial Intelligence, July 2-6, Paris. Sponsor: AFCET. Contact J.-L. Le Moigne, GRASCE, Univ. Aix Marseille III, 3, ave. Robert Schuman, 13628, Aix en Provence, France; or P. Bourguin, 26 rue St. Louis, 78000, Versailles, France.

SPAA 90, Second ACM Symp. on Parallel Algorithms and Architecture, July 2-6, Crete, Greece. Cosponsor: ACM. Contact Tom Leighton, MIT, Math Dept. and Computer Science Lab, Cambridge, MA 02139, phone (617) 253-3662, e-mail tli@math.mit.edu.

Fourth TC2 Working Conf. on Database Semantics, July 2-6, Windermere Lake District, UK. Sponsor: IFIP, Coopers and Lybrand UK. Contact William Kent, Hewlett-Packard Labs, Dept. 3U, 1501 Page Mill Rd., Palo Alto, CA 94304-0971; or Robert Meersman, Infolab, Tilburg Univ., PO Box 90153, 5000 LE Tilburg, The Netherlands.

Fifth IEEE Structure in Complexity Theory Conf., July 7-11, Barcelona, Spain. Contact Stephen R. Mahaney, Computer Science Dept., Univ. of Arizona, Gould-Simpson 721, Tucson, AZ 85721, phone (602) 621-2733.

WCCE 90, Fifth World Conf. on Computers in Education, July 9-13, Sydney, Australia. Cosponsors: IFIP et al. Contact WCCE 90, PO Box 319, Darlinghurst, NSW 2010, Australia, phone (612) 211-5855.

Iberamerica 90, Second Ibero-American Conf. on AI, July 9-13, Morelia, Michoacan, Mexico. Sponsors: Centro Regional de Enseñanza en Informática (Spain) et al. Contact Iberamerica 90, Atm. Srita. Ma. Antonieta Alvarez Perez, Apartado Postal 70302, C.P. 04510, Mexico, D.F.

Int'l Neural Network Conf., July 9-13, Paris. Sponsors: IEEE Neural Networks Council, Int'l Neural Network Society. Contact Nina Thellier, NTC, 19, rue de la Tour, 75116 Paris Cedex, France, phone (33) 45-25-65-65, fax (33) 45-25-24-22.

 Third Int'l Conf. on Industrial and Engineering Applications for AI and Expert Systems, July 15-18, Charleston, S.C. Sponsors: ACM et al. Contact Moonis Ali, Univ. of Tennessee Space Inst., MS15, Tullahoma, TN 37388, phone (615) 455-0631.

1990 Summer Computer Simulation Conf., July 16-18, Calgary, Alta., Canada. Sponsor: Society for Computer Simulation. Contact SCS, PO Box 17900, San Diego, CA 92117-7900, phone (619) 277-3888.

ICALP 90, 17th Int'l Colloquium on Automata, Languages, and Programming, July 16-20, Coventry, England. Sponsor: Int'l Computers, Ltd. Contact Computer Science Dept., Univ. of Warwick, Coventry CV4 7AL, UK, phone 44 (203) 523-194.

Int'l Workshop on Semantics for Concurrency, July 23-25, Leicester, UK. Sponsor: British Computer Society, Contact Maria Kwiatkowska, Workshop on Semantics for Concurrency, Computing Studies Dept., Univ. of Leicester, Leicester LE1 7RH, UK, phone (44) 533-523603.

Int'l Workshop on Principles of Diagnosis, July 23-25, Menlo Park, Calif. Cosponsors: American Assoc. for Artificial Intelligence, Price Waterhouse. Contact Walter Hamscher, Price Waterhouse Technology Center, 68 Willow Rd., Menlo Park, CA 94025, phone (415) 688-6669, e-mail hamscher@pw.com or wh@ai.ai.mit.edu.

10th Int'l Conf. in Computer Science, July 23-27, Santiago, Chile. Contact Joachim von zur Gathen, Computer Science Dept., Univ. of Toronto, 10 King's College Rd., Toronto, Canada M5S 1A4, phone (416) 978-6024, e-mail gathen@theory.toronto.edu.

DIAC 90, Directions and Implications of Advanced Computing, July 28, Boston. Sponsor: Computer Professionals for Social Responsibility. Contact Douglas Schuler, Boeing Computer Services, MS 7L-64, PO 24346, Seattle, WA 98124-0346, phone (206) 634-2771.

AAAI 90 Workshop of the Nat'l Conf. on AI, July 31-Aug. 3, Boston. Sponsor: American Assoc. for Artificial Intelligence. Contact Edward Lafferty, AI Center, Mire, MS A350, Burlington Rd., Bedford, MA 01730, phone (617) 271-2773.

August 1990

 SIGGraph 90, 17th Conf. on Computer Graphics and Interactive Techniques, Aug. 6-10, Dallas. Sponsor: ACM. Contact Assoc. for Computing Machinery, 11 W. 42nd St., New York, NY 10036, phone

(212) 869-7440; or SIGGraph 90, 111 E. Wacker Dr., Suite 600, Chicago, IL 60601, fax (312) 938-1232.

 16th Int'l Conf. on Very Large Data Bases, Aug. 13-16, Brisbane, Australia. Contact David Reiner, Lotus Development, 1 Canal Park, Cambridge, MA 02141, phone (617) 577-8500, e-mail dreiner@lotus.com.

ICPP 90, 19th Int'l Conf. on Parallel Processing, Aug. 13-17, St. Charles, Ill. Sponsor: Pennsylvania State Univ. Contact Tse-yun Feng, EE East Bldg., Pennsylvania State Univ., University Park, PA 16802, phone (814) 863-1469.

TAU 90, 1990 Int'l Workshop on Timing Issues in the Specification and Synthesis of Digital Systems, Aug. 15-17, Vancouver, B.C., Canada. Sponsor: ACM. Contact Robert K. Brayton, Electrical Engineering and Computer Science Dept., Univ. of California at Berkeley, Berkeley, CA 94760.

Int'l Symp. on Algorithms, Aug. 16-18, Tokyo. Sponsor: IPSJ Special Interest Group on Algorithms. Contact Tetsuo Asano, Osaka Electro-Communication Univ., Hatsu-cho, Neyagawa, Osaka 572, Japan, phone 81 (720) 24-1131.

UPADI 90, 21th Convention of the Pan American Federation of Engineering Societies, Aug. 19-24, Washington, DC. Cosponsors: American Assoc. of Engineering Societies, American Society of Civil Engineers. Contact UPADI 90, ASCE, 345 E. 47th St., New York, NY 10017, phone (212) 705-7218.

 Hot Chips 2 Symp., Aug. 20-21, Santa Clara, Calif. Contact Hasan S. Alkhataib, EECS Dept., Santa Clara Univ., Santa Clara, CA 95053, phone (408) 554-4485, fax (408) 554-5475, e-mail halkhatib@scu.edu.

Second Int'l Joint Conf. of ISSAC 90 (1990 Int'l Symp. on Symbolic and Algebraic Computation) and AAECC 8 (Eighth Int'l Conf. on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes), Aug. 20-24, Tokyo. Cosponsors: ACM et al. Contact Conf. Secretariat, IJC-2, Scientist, Inc., Yamazaki Bldg., 3-2 Kanda Suruga-dai, Chiyoda-ku, Tokyo 101, Japan.

Coling 90, 13th Int'l Conf. on Computational Linguistics, Aug. 20-25, Helsinki, Finland. Contact Hans Karlgen, KVAL, Skeppsbron 26, S-111 30 Stockholm, Sweden, phone 46 (8) 789-6683.

September 1990

 ISPRS Commission V Symp., Close-Range Photogrammetry Meets Machine Vision, Sept. 3-7, Zurich. Sponsor: Int'l Society for Photogrammetry and Remote Sensing et al. Contact Armin Gruen, Inst. of Geodesy and Photogrammetry, ETH-Hoenggerberg, CH-8093, Zurich, Switzerland, phone 41 (1) 377-3051.

 EuroVHDL 90, First European Working Conf. on VHDL Methods, Sept. 4-7, Marseille, France. Cosponsors: ACM et al. Contact Peter Michel, Siemens, A.G. Dept. ZFE ISEAI, Otto Hahn Ring 6, Munich 83, West Germany.

ASAP 90, Int'l Conf. on Application-Specific Array Processors, Sept. 5-7, Princeton, NJ. Cosponsor: Princeton Univ. Contact S.Y. Kung, Electrical Engineering Dept., Princeton Univ., Princeton, NJ 08544, phone (609) 258-3780.

13th Int'l ACM/SIGIR Conf. on Research and Development in Information Retrieval, Sept. 5-7, Brussels. Contact Jean-Luc Vidick, Univ. Libre de Bruxelles, Avenue F.D. Roosevelt, Infocorp, C.P. 142, 1050 Brussels, Belgium.

Int'l Workshop on VLSI for Artificial Intelligence and Neural Networks, Sept. 5-7, Oxford, England. Contact Jose G. Delgado-Frias, Electrical Engineering Dept., SUNY, Binghamton, NY 13901, phone (607) 777-4806, e-mail delgado@bingvaxu.cc.binghamton.edu.

1990 Int'l Electronics Packaging Conf., Sept. 9-13, Marlborough, Mass. Sponsor: Int'l Electronics Packaging Society. Contact IEPS, 114 N. Hale St., Wheaton, IL 60187, phone (708) 260-1044.

Workshop on Computers in Systematic Biology, Sept. 9-14, Davis, Calif. Sponsor: Nat'l Science Foundation. Contact Renaud Fortner, California Dept. of Food and Agriculture, Analysis and Identification, Rm. 340, PO Box 942871, Sacramento, CA 94271-0001, phone (916) 445-4521.

 ITC 90, Int'l Test Conf., Sept. 10-12, Washington, DC. Cosponsor: IEEE Philadelphia Section. Contact Donald Denburg, AT&T Bell Labs, 1247 S. Cedar Crest Blvd., Allentown, PA 18103; or ITC, 1201 Sussex Turnpike, Suite 101, PO Box 264, Mt. Freedom, NJ 07970, phone (201) 895-5260.

 IEEE Conf. on Managing Expert System Programs and Projects, Sept. 10-12, Washington, DC. Sponsor: IEEE Computer Society Technical Committee on Expert Systems. Contact Jay Liebowitz, Management Sciences Dept., George Washington Univ., Washington, DC, phone (202) 994-6969.

Second Int'l Workshop on Advances in Robot Kinematics, Sept. 10-12, Linz, Austria. Sponsors: Research Inst. for Symbolic Computation et al. Contact Sabine Stifter, RISC, Johannes Kepler Univ., A-4040 Linz, Austria, phone 43 (7236) 3231-50; or Jadran Lenaric, Josef Stefan Inst., Univ. of Edvard Kardelj, Jovova 39, 61111 Ljubljana, Yugoslavia, phone 38 (61) 214-399.

Symp. on Object-Oriented Programming Emphasizing Practical Applications, Sept. 14-15, Poughkeepsie, N.Y. Sponsor: Marist College. Contact James TenEyck, Marist College, Poughkeepsie, NY 12601-1387, phone (914) 471-3240, e-mail jtbv@maristb.bitnet.

ICCD 90, IEEE Int'l Conf. on Computer Design, VLSI in Computers and Processors, Sept. 16-19, Cambridge, Mass. Contact Edward M. Middleworth, Hewlett-Packard, Bldg. 25U, PO Box 10350, Palo Alto, CA 94303-0867, phone (415) 857-5485; or ICCD 90, IEEE Computer Society, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

Fourth Digital Signal Processing Workshop, Sept. 16-19, New Paltz, N.Y. Sponsor: IEEE Signal Processing Society. Contact K.S. Arun, Coordinated Science Lab, Univ. of Illinois at Urbana-Champaign, 1101 W. Springfield Ave., Urbana, IL 61801, phone (217) 333-7678, fax (217) 244-1764.

Internal Audit Advanced Technology Forum, Sept. 17-19, Orlando, Fla. Sponsor: Inst. of Internal Auditors. Contact Stephen M. Paroby, Ernst and Young, 787 Seventh Ave., New York, NY 10019, phone (212) 830-6000.

ASIC 90, Third IEEE ASIC Seminar and Exhibit, Sept. 17-21, Rochester, N.Y. Cosponsors: IEEE Rochester Section, ACM. Contact Kenneth Hsu, Rochester Inst. of Technology, Computer Engineering Dept., Rochester, NY 14623, phone (716) 475-2655; or Lynne Engelbrecht, 170 Mt. Read Blvd., Rochester, NY 14611, phone (716) 328-2310, fax (716) 436-9370.

EP 90, Electronic Publishing '90, Sept. 18-20, Gaithersburg, Md. Sponsor: Nat'l Inst. of Standards and Technology. Contact Peter R. King, Computer Science Dept., Univ. of Manitoba, Winnipeg, Man., Canada R3T 2N2, phone (204) 474-9935.

ICARCV 90, Int'l Conf. on Automation, Robotics, and Computer Vision, Sept. 18-21, Singapore. Cosponsors: IEEE Singapore Chapter et al. Contact Dinesh P. Mital, ICARCV 90, School of Electrical and Electronic Engineering, Nanyang Technological Inst., Nanyang Ave., Singapore 2263, Republic of Singapore, phone (65) 660-5399.

Conf. on Multiuser Interfaces and Applications, Sept. 24-26, Heraklion, Crete, Greece. Cosponsors: IFIP et al. Contact Nera Kalaitzaki, Computer Science Dept., Univ. of Crete, GR 714-09 Heraklion, Crete, Greece, phone (81) 210-057.

Int'l Workshop on Expert Systems in Engineering, Sept. 24-26, Vienna, Austria. Sponsor: Christian Doppler Expert Systems Lab, Univ. of Vienna. Contact Wolfgang Nejdl, Technical Univ. of Vienna, Applied Computer Science Dept., CD Lab for Expert Systems, Panigasse 16, 1040 Vienna, Austria, fax (43) (222) 505-5304, e-mail nejdl@uxpert.at.

Tencon 90, IEEE Region 10 Conf. on Computer and Communication Systems, Sept. 24-27, Hong Kong. Cosponsor: IEEE Hong Kong Section. Contact Y.S. Cheung, Electrical and Electronic Engineering Dept., Univ. of Hong Kong, Pokfulam, Hong Kong.

SIGComm 90, Sept. 24-27, Philadelphia. Sponsor: ACM SIGComm. Contact David

Farber, Univ. of Pennsylvania, 200 S. 33rd St., Philadelphia, PA 19104-6389, phone (215) 898-9508, fax (215) 898-0587, e-mail farber@cis.upenn.edu; or Phil Kam, Bell Communications Research, MS 2P-357, 445 South St., PO Box 1910, Morristown, NJ 07962-1910, phone (201) 829-4299.

AIRIES 90, AI Research in the Environmental Sciences Workshop, Sept. 25-27, Montreal, Que., Canada. Cosponsors: Univ. of Quebec at Montreal, Centre Recherche Informatique de Montreal. Contact Rosemary M. Dyer, GL/LLP, AIRIES 90, Air Force Geophysics Lab, Hanscom Air Force Base, MA 01731, fax (617) 377-4498.

Fourth Conf. on Putting Methods and Tools into Practice as Aids to Design Information Systems, Sept. 25-27, Nantes, France. Sponsor: Univ. of Nantes, Inst. Univ. de Technologie, Lab. d'Informatique, Liana. Contact H. Habrias, 3 Rue du Marechal Joffre, 44041 Nantes Cedex 01, France, phone (33) 4030-6090, fax (33) 4030-6001.

CI 90, 1990 Int'l Symp. on Computational Intelligence, Sept. 27-29, Milano, Italy. Sponsors: ACM, F.I.S. Cassa di Risp. o. PC. Contact Giorgio Valle, Università Milano. Dip. Scienze Della Informazione, Via Moretto 2233, Milano, Italy, phone 39 (2) 757-5218, fax 39 (2) 761-10556, e-mail valle@imiucca.bitnet.

October 1990

THIR UNB Artificial Intelligence Workshop, Oct. 1, Fredericton, N.B., Canada. Sponsor: Univ. of New Brunswick. Contact B.G. Nickerson, School of Computer Science, Univ. of New Brunswick, PO Box 4400, Fredericton, N.B., Canada E3B 5A3, phone (506) 453-4566, fax (506) 453-3566, e-mail BGN@UNB.CA.

15th Conf. on Local Computer Networking, Oct. 1-3, Minneapolis, Minn. Contact Marc Cohn, Advanced Development Div., Raychem Corp., 300 Constitution Dr., Menlo Park, CA 94025-1164, phone (415) 361-3902, fax (415) 361-6099.

Second Int'l Conf. on Algebraic and Logic Programming, Oct. 1-3, Nancy, France. Contact Wolfgang Wechler, TU Braunschweig, Theoretische Informatik, Postfach 3329, D-3300 Braunschweig, West Germany, e-mail wechler@infs.uucp; or Helene Kirchner, CRIN, BP239, 54506 Vandoeuvre-les-Nancy Cedex, France.

InfoJapan 90, Int'l Conf. on Information Technology, Oct. 1-5, Tokyo. Sponsor: IPSJ. Contact Takuma Yamamoto, Fujitsu, 3-14-1 Hiyoshi, Kokoku-ku, Yokohama, Japan.

Sixth Int'l Conf. on the Application of Standards for Open Systems Interconnection, Oct. 2-4, Gaithersburg, Md. Cosponsor: Nat'l Inst. of Standards and Technology.

Contact Brenda Gray, NIST/OSI, Rm. B217, Bldg. 225, Gaithersburg, MD 20899, phone (301) 975-3664.

28th Allerton Conf. on Communication, Control, and Computing, Oct. 3-5, Monticello, Ill. Contact Allerton Conf., c/o Donna J. Brown, Univ. of Illinois at Urbana-Champaign, Coordinated Science Lab, 1101 W. Springfield Ave., Urbana, IL 61801, phone (217) 244-0581, e-mail djb@uicis.cs.uiuc.edu.

1990 IEEE Workshop on Visual Languages, Oct. 4-6, Skokie, Ill. Cosponsors: Univ. of Pittsburgh et al. Contact S.K. Chang, Computer Science Dept., Univ. of Pittsburgh, Pittsburgh, PA 15260.

Frontiers 90, Third Symp. on Frontiers of Massively Parallel Computation, Oct. 8-10, College Park, Md. Cosponsor: NASA Goddard Space Flight Center. Contact Johanna Weinstein, Frontiers 90, UMIACS, Univ. of Maryland, A.V. Williams Bldg., College Park, MD 20742, phone (301) 454-1808.

Future Trends '90, Workshop on Future Trends of Distributed Computing Systems, Oct. 8-10, Cairo. Contact Stephen S. Yau, Univ. of Florida, CIS Dept., Rm. 301, Gainesville, FL 32611, phone (904) 335-8006.

Ninth Symp. on Reliable Distributed Systems, Oct. 9-11, Huntsville, Ala. Contact Raif M. Yanney, TRW, MS DH2/2328, 1 Space Park, Redondo Beach, CA 90278, phone (213) 764-6033.

Northcon 90, Oct. 9-11, Seattle. Cosponsors: IEEE et al. Contact Northcon 90 Professional Program Committee, c/o Ramona Baker, 8110 Airport Blvd., Los Angeles, CA 90045-3194, phone (213) 215-3796, ext. 222.

PDCS 90, ISMM Int'l Conf. on Parallel and Distributed Computing and Systems, Oct. 10-12, New York City. Sponsor: Int'l Society for Mini and Microcomputers. Contact R. Ammar, U155, Computer Science and Engineering Dept., Univ. of Connecticut, Storrs, CT 06268, fax (203) 486-0318.

EuroForum 90, Fourth European EDIF Forum, Oct. 11-12, Daresbury, Cheshire, UK. Contact Kate Faulkner, EuroForum 90, ICL, Manchester M12 5DR, UK phone 44 (61) 223-1301, fax 44 (61) 223-1207.

Second Int'l Conf. on Microelectronics, Oct. 13-15, Damascus, Syria. Sponsor: Arab School of Science and Technology. Contact M.I. Elmasy, VLSI Research Group, Univ. of Waterloo, Waterloo, Ont., Canada N2L 3G1, phone (519) 885-1211, ext. 3753.

19th AIPR Workshop on Applied Imagery Pattern Recognition, Oct. 17-19, McLean, Va. Sponsors: Society of Photocopying Instrumentation Engineers, Rome Air Development Center. Contact Brian Mitchell, ERIM, PO Box 8618, Ann Arbor, MI 48106, phone (313) 994-1200, ext. 2713.

Third Fall VHDL Users' Group Meeting, Oct. 21-24, Redondo Beach, Calif.

Contact Rachel Rusting, Intermetrics, 733 Concord Ave., Cambridge, MA 02138, phone (617) 661-1840.

12th Saudi Nat'l Computer Conf. on Planning for the Informatics Society, Oct. 21-24, Riyadh, Saudi Arabia. Cosponsors: King Saud Univ., Saudi Computer Society, Contact Mohammad M. Mandurah, College of Computer and Information Sciences, PO Box 51178, Riyadh, 11543, Kingdom of Saudi Arabia, phone 996 (1) 467-6993.

OOPLSA 90, Fifth Conf. on Object-Oriented Programming Systems, Languages, and Applications, Oct. 21-25, Ottawa, Canada. Sponsor: ACM. Contact Assoc. for Computing Machinery, 11 W. 42nd St., New York, NY 10036, phone (212) 869-7440.

FOCS, 31st Foundations of Computer Science, Oct. 22-24, St. Louis, Mo. Contact Christos Papadimitriou, Computer Science Dept., Univ. of California at San Diego, La Jolla, CA 92093, phone (619) 534-2086.

Int'l Conf. on Computer Applications in Developing Countries, Oct. 22-24, Benin City, Nigeria. Sponsor: Large Scale Systems Research Group, Univ. of Benin. Contact E.A. Onibere, Mathematics and Computer Science Dept., Univ. of Benin, P.M.B. 1154, Benin City, Nigeria.

Ninth National Conf. on EDP System and Software Quality Assurance, Oct. 22-24, Washington, DC. Sponsor: Data Processing Management Assoc. Contact US Professional Development Inst., EDP System and Software Quality Assurance, 1734 Elton Rd., Suite 221, Silver Spring, MD 20903-1733, phone (301) 445-4400, fax (301) 445-5722.

JCIT 5, Fifth Jerusalem Conf. on Information Technology, Oct. 22-25, Jerusalem, Israel. Sponsor: Information Processing Assoc. of Israel. Contact Abraham Peled, IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, NY 10598.

CC 90, Third Int'l Workshop on Compiler Compilers, Oct. 22-26, Schwerin, East Germany. Sponsors: German Democratic Republic Academy of Sciences Inst. of Informatics and Computing Technique e.g. Contact Michael Albinus, CC 90 Organizing Committee, Akademie der Wissenschaften der DDR, Inst. für Informatik und Rechenstechnik, Rudower Chaussee 5, Berlin, GDR — 1199.

Third Int'l Symp. on Artificial Intelligence, Oct. 22-26, Monterrey, N.L. Mexico. Sponsors: ITESM (Inst. Tecnológico y de Estudios Superiores de Monterrey) e.g. Contact Hugo Terashima, Centro de Inteligencia Artificial, ITESM, Suc. de Correo "J", C.P. 64849 Monterrey, N.L. Mexico, phone 52 (83) 58-2000, fax 52 (83) 58-0771, e-mail isa@tecmy vm.bitnet.

Visualization 90, Oct. 23-26, San Francisco. Contact Bruce Brown, Oracle Corp., 20 Davis Dr., Belmont, CA 94002, phone (415) 598-3628.

ESORICS 90, European Symp. on Research in Computer Security, Oct. 24-26, Toulouse, France. Sponsor: AFCEC. Contact Martin Gilles, 16 Para de Diane, 78350 Jouy eu Jossas, Toulouse Cedex, France.

First Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop, Oct. 25-26, Kyoto, Japan, and Oct. 29-31, Tokyo. Cosponsors: Kansai Inst. of Information Systems e.g. Contact John H. Boose, Advanced Technology Center, Boeing Computer Services 7L-64, PO Box 24346, Seattle, WA 98124, phone (206) 865-3253.

NACLP 90, 1990 North American Conf. on Logic Programming, Oct. 28-Nov. 1, Austin, Texas. Cosponsor: ACM. Contact Carlo Zaniolo, MCC, 3500 W. Balcones Center Dr., Austin, TX 78759, phone (512) 338-3442.

Int'l Conf. on Information Technology, Oct. 29-31, Bournemouth, UK. Sponsor: Institution of Electrical Engineers. Contact IEE Conf. Services, Savoy Pl., London WC2R 0BL, UK, phone 44 (1) 24-01-871.

Eighth Pacific Northwest Software Quality Conf., Oct. 29-31, Portland, Ore. Sponsor: PNSQC Committee. Contact Terri Moore, Pacific Agenda, PO Box 10142, Portland, OR 97210, phone (503) 223-8633.

ISCSIA 5, Fifth Int'l Symp. on Computer and Information Sciences, Oct. 30-Nov. 2, Cappadocia, Nevşehir, Turkey. Sponsors: Istanbul Technical Univ. e.g. Contact A. Emre Harmanci, Istanbul Technical Univ., Bilgi İslam Merkezi, Ayazaga, 80626 Istanbul, Turkey, phone 090 (1) 176-3254, fax 090 (1) 176-1734, e-mail harmanci@trluu.bitnet.

Compsac 90, 14th Int'l Computer Software and Applications Conf., Oct. 31-Nov. 2, Chicago. Contact Ifay F. Chang, Rm. 1B28, IBM T.J. Watson Research Center, PO Box 714, Yorktown Heights, NY 10595, phone (914) 789-7825.

November 1990

14th SCAMC, 1990 Symp. on Computer Applications in Medical Care, Nov. 4-7, Washington, DC. Cosponsors: George Washington Univ. Medical Center e.g. Contact SCAMC — Office of CEM, George Washington Univ. Medical Center, 2300 K St. NW, Washington, DC 20037, phone (202) 994-8928.

24th Asilomar Conf. on Signals, Systems, and Computers, Nov. 5-7, Pacific Grove, Calif. Sponsors: Naval Postgraduate School e.g. Contact George M. Dillard, Naval Ocean Systems Center, San Diego, CA 92152-0000, phone (619) 553-2478.

1990 IFIP-IEEE Int'l Workshop on Defect and Fault Tolerance in VLSI Systems, Nov. 5-7, Grenoble, France. Contact Gabriel Sau-

cier, Inst. National Polytechnique de Grenoble/CSL, 46, avenue Felix-Viallet, 38031 Grenoble Cedex, France, phone (33) 76-57-46-87, fax (33) 76-50-23-21, or Tulin E. Mangir, TRW, 1 Space Park, R2/2036, Redondo Beach, CA 90278, phone (213) 813-3894, fax (213) 813-3709.

ICCS 90, Int'l Conf. on Communication Systems, Nov. 5-9, Singapore. Cosponsors: IEEE Singapore Section e.g. Contact ICCS 90, c/o Meeting Planners Pte. Ltd., 100 Beach Rd. #33-01, Shaw Towers, Singapore 0718.

Second SIAM Conf. on Linear Algebra in Signals, Systems, and Control, Nov. 5-9, San Francisco, Calif. Sponsor: Society for Industrial and Applied Mathematics. Contact SIAM, 3600 University City Science Center, Philadelphia, PA 19104-2688, phone (215) 382-9800, fax (215) 386-7999, e-mail siam@wharton.upenn.edu.

ICCC 90, 10th Int'l Conf. on Computer Communication, Nov. 5-9, New Delhi, India. Sponsor: Int'l Council on Computer Communication. Contact Saroj Chowla or P.P. Gupta, ICC 90, CMC Ltd., A-5 Ring Rd., South Extension Part I, New Delhi 110 049, India, phone 91 (11) 626-8077, fax 91 (11) 684-4652.

Intelligent Robotic Systems: Design and Applications, Nov. 6-7, Philadelphia. Sponsor: SPIE. Contact Mohan M. Trivedi, Univ. of Tennessee, Electrical and Computer Engineering, Ferris Hall, Knoxville, TN 37996-2100, phone (615) 974-5450.

TAI 90, Second Computer Society Int'l Conf. on Tools for Artificial Intelligence, Nov. 6-9, Washington, DC. Cosponsors: Rutgers Univ. e.g. Contact Nicholas G. Bourbakis, George Mason Univ., ECE Dept., Fairfax, VA 22030, phone (703) 425-3930.

IEEE Workshop on the Management of Replicated Data, Nov. 7-9, Houston. Sponsor: IEEE Technical Committee on Operating Systems. Contact Jehan-Francois Paris, Computer Science Dept., Univ. of Houston, Houston, TX 77204-3475, phone (713) 749-3943, e-mail paris@cs.uh.edu; or Luis-Felipe Cabrera, IBM Almaden Research Center, 650 Harry Rd., MC K55/803, San Jose, CA 95120-6099, phone (408) 927-1838.

1990 IEEE Workshop on VLSI Signal Processing, Nov. 7-9, San Diego, Calif. Contact Patti Fenstermacher, AT&T Bell Labs, 1243 S. Cedar Crest Blvd., Allentown, PA 18103, e-mail p8@aolatt.com; or Howard S. Moscovitz, AT&T Bell Labs, 1243 S. Cedar Crest Blvd., Allentown, PA 18103, e-mail mosc@aolatt.com.

ICCAD 90, IEEE Int'l Conf. on Computer-Aided Design, Nov. 11-15, Santa Clara, Calif. Cosponsors: IEEE Circuits and Systems Society. Contact Pat Pistilli, MP Associates, 7490 Clubhouse Dr., Suite 102, Boulder, CO 80301, phone (303) 530-4562 or 4333.

Supercomputing 90, Nov. 12-16, New York City. Cosponsor: ACM. Contact Joanne L. Martin, IBM T.J. Watson Research

Center, PO Box 218, Route 134, Yorktown Heights, NY 10698, phone (914) 945-3285, e-mail jlmart@ibm.com; or Supercomputing 90, IEEE Computer Society, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

Seventh Governor's Symp. on High Technology, Nov. 13-15, Kauai, Hawaii. Sponsor: State of Hawaii. Contact William M. Ball, State of Hawaii, 300 Kahalo St., Suite 35, Mililani, HI 96789, phone (808) 625-5293.

PRICAI 90, Pacific Rim Int'l Conf. on Artificial Intelligence, Nov. 14-16, Nagoya-shi, Aichi, Japan. Sponsor: Japanese Society for Artificial Intelligence et al. Contact Tetsuo Fukumura, Inter Group Corp., Akasaka Yamakatsu Bldg., 8-5-32 Akasaka, Minato-ku, Tokyo 107, Japan, phone (03) 479-5535.

Cognitiva 90, Nov. 20-23, Madrid. Sponsor: AFCEC. Contact Cognitiva 90, c/o AFCEC, 156 Bd. Pereire, 75017 Paris, France, phone (33) (01) 47-66-24-19.

IEEE 1990 Conf. on Software Maintenance, Nov. 26-29, San Diego, Calif. Contact Thomas M. Pigowski, USNSG Pensacola, Corry Station, Pensacola, FL 32511, phone (904) 452-6399.

Micro 23, 23rd Symp. and Workshop on Microprogramming and Microarchitecture, Nov. 27-29, Orlando, Fla. Cosponsor: ACM. Contact Chris Papachristou, Case Western Reserve Univ., Computer Engineering and Science Dept., Cleveland, OH 44106, phone (216) 368-5277, e-mail cap@alpha.ces.cwru.edu.

December 1990

First Int'l Symp. on Uncertainty and Analysis: Fuzzy Reasoning, Probabilistic Methods, and Risk Management, Dec. 3-5, College Park, MD. Sponsors: Univ. of Maryland et al. Contact Bilal M. Ayub, Civil Engineering Dept., Univ. of Maryland, College Park, MD 20742.

ACM SIGSOFT 90, Fourth Symp. on Software Development Environments, Dec. 3-5, Irvine, Calif. Sponsor: ACM. Contact Dewayne E. Perry, AT&T Bell Labs, 600 Mountain Ave., Murray Hill, NJ 07974, phone (201) 582-2529.

Sixth Computer Security Applications Conf., Dec. 3-7, Tucson, Ariz. Sponsors: American Society for Industrial Security et al. Contact Marshall D. Abrams, Mitre Corp., 7525 Colshire Dr., M/S 2269, McLean, VA 22101, phone (703) 883-6938, e-mail abrams@mitre.org.

ICCV 90, Third Int'l Conf. on Computer Vision, Dec. 4-7, Osaka, Japan. Contact ICCV 90, IEEE Computer Society, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

11th Real-Time Systems Symp., Dec. 5-7, Orlando, Fla. Sponsor: IEEE Computer Society Technical Committee on Real-Time Computing. Contact Doug Locke, IBM — MS 409, Systems Integration Div., 6600 Rockledge Dr., Bethesda, MD 20817, phone (301) 493-1496, e-mail cdl@cs.cmu.edu.

CASE 90, Fourth Int'l Workshop on Computer-Aided Software Engineering, Dec. 5-8, Irvine, Calif. Contact Elliott J. Chikofsky, Radius Systems, 75 Lexington St., Burlington, MA 01803, phone (617) 494-8200.

San Diego Workshop on Volume Visualization, Dec. 10-11, La Jolla, Calif. Cosponsor: ACM. Contact T. Todd Elvins, SDSC, Box 85608, San Diego, CA 92038, phone (619) 534-5128.

Second IEEE Symp. on Parallel and Distributed Processing, Dec. 10-12, Dallas. Cosponsor: Dallas Chapter of the IEEE Computer Society. Contact Behrooz Shirazi, Computer Science Dept., Southern Methodist Univ., 6425 Airline Rd., Dallas, TX 75205-2337, phone (214) 692-2874, e-mail shirazi@smu.uucp@uunet.uu.net.

1990 IEEE Workshop on Languages and Architectures for Automation, Dec. 19-21, Honolulu, Hawaii. Sponsors: IEEE Int'l Center for High Technology Research et al. Contact D.Y.Y. Yun, Univ. of Hawaii, 711 Kapiolani Blvd., Suite 200, Honolulu, HI 96813-5249, phone (808) 539-1532, fax (808) 941-1399; or Shi-Kuo Chang, 322 Alumni Hall, Univ. of Pittsburgh, Pittsburgh, PA 15260, phone (412) 624-8493, fax (412) 624-8465, e-mail chang@vax.cs.pitt.edu.

January 1991

Fourth CSI/IEEE Int'l Symp. on VLSI Design, Jan. 5-8, New Delhi. Sponsors: Computer Society of India et al. Contact Yashwant K. Malaiya, Computer Science Dept., Colorado State Univ., Fort Collins, CO 80523, phone (303) 491-7031, fax (303) 491-2293, e-mail malaiya@ravi.cs.colostate.edu; or D. Roy Chowdhury, Gateway Design Automation, SDF/A-1, Noida Export Processing Zone 10 NEPZ, Noida 201305, India, phone 91 (05736) 62342, fax 91 (05736) 62343.

Int'l Conf. on Multimedia Information Systems, Jan. 16-18, Singapore. Contact Juzar Motiwalla, Inst. of Systems Science, Nat'l Univ. of Singapore, Heng Mui Keng Terr., Kent Ridge, Singapore 0511, phone (65) 772-2075.

PADS, Workshop on Parallel and Distributed Simulation, Jan. 21-23, Anaheim, Calif. Cosponsors: ACM, SCS. Contact David M. Nicol, Computer Science Dept., College of William and Mary, Williamsburg, VA 23185, phone (804) 221-3458, e-mail nicol@cs.wm.edu.

February 1991

CAIA 91, Seventh IEEE Conf. on Artificial Intelligence Applications, Feb. 24-28, Miami Beach, Fla. Contact IEEE Computer Society, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

Compscon Spring 91, Feb. 25-Mar. 1, San Francisco. Contact Compscon Spring 91, IEEE Computer Society, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

March 1991

Fifth Int'l Workshop on High-Level Synthesis, Mar. 3-6, Buhlerhoh, West Germany. Cosponsors: IEEE et al. Contact Raul Camposano, IBM T.J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598, phone (914) 945-3871, e-mail raul@ibm.com.

CEEDA 91, Int'l Conf. on Concurrent Engineering and Electronic Design Automation, Mar. 26-28, Bournemouth, Dorset, UK. Contact S. Medhat, Dorset Inst., Electronic and Engineering Dept., Wallisdown Road, Wallisdown Poole BH125BB, UK, phone (02022) 595-494.

10th IEEE Int'l Phoenix Conf. on Computers and Communications, Mar. 27-30, Scottsdale, Ariz. Sponsors: IEEE, IEEE Communications Society. Contact Oris Friesen, Bull HN, PO Box 8000, MS A93, Phoenix, AZ 85066, phone (602) 862-5200, e-mail Friesen@system-m.phx.bull.com.

April 1991

First IEEE Int'l Workshop on Interoperability in Multidatabase Systems, Apr. 8-9, Kyoto, Japan. Contact Ahmed K. Elmagarmid, Purdue Univ., Computer Sciences Dept., West Lafayette, IN 47907, phone (317) 494-1998; or Yutaka Matsushita, Instrumentation Dept., Yokota Univ., Hiroyoshi, Yokohama, Japan, phone 81 (44) 63-1141, ext. 3564.

Seventh Int'l Conf. on Data Engineering, Apr. 8-12, Kobe, Japan. Contact Ming T. (Mike) Liu, Computer and Information Science Dept., Ohio State Univ., 2036 Neil Ave., Columbus, OH 43210-1277, phone (614) 292-1837, e-mail Lliu@CIS.IRCC. Ohio State Univ.; or Data Engineering 91, IEEE Computer Society, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013, fax (202) 728-0884.

ETC 91, 1991 European Test Conf., Apr. 17-19, Munich, West Germany. Sponsors: VDE (Zentralstelle Tagungen und Seminare), Contact Peter Stiklie, VDE, Stresemannallee 15, D-6000 Frankfurt 70, West Germany, phone (69) 6308-203, fax (69) 6308-273.

CALL FOR PAPERS

Tri-Ada 90: Dec. 3-7, 1990, Baltimore, Md. Sponsor: ACM. Submit six copies of abstract/paper by **May 21, 1990**, to Erhard Ploedereder, Tartan Labs, 300 Oxford Dr., Monroeville, PA 15146, phone (412) 856-3600, fax (412) 856-3636, e-mail ploeder@tartan.com or ploeder@ajpo.sci.cmu.edu.

15th Conf. on Local Computer Networks: Oct. 1-3, 1990, Minneapolis, Minn. Submit five copies of paper by **May 22, 1990**, to Marc Cohn, Advanced Development Div., Raychem Corp., 300 Constitution Dr., Menlo Park, CA 94025-1164, phone (415) 361-3902, fax (415) 361-6099.

Symp. on Object-Oriented Programming Emphasizing Practical Applications: Sept. 14-15, 1990, Poughkeepsie, N.Y. Sponsor: Marist College. Submit three copies of paper or panel proposal by **May 31, 1990**, to Fred Benfer, IBM Corporate Education, Thorwood, NY 10594, e-mail hzap@maristb.bitnet.

24th Asilomar Conf. on Signals, Systems, and Computers: Nov. 5-7, 1990, Pacific Grove, Calif. Sponsors: Naval Postgraduate School et al. Submit three copies of abstract and extensive summary by **June 1, 1990**, to Frederic J. Harris, Electrical and Computer Engineering Dept., San Diego State Univ., San Diego, CA 92182-0190, phone (619) 594-6162.

Seventh Israeli Conf. on Artificial Intelligence and Computer Vision: Dec. 26-27, 1990, Tel Aviv. Submit four copies of full paper by **June 1, 1990**, to A. Bruckstein, Faculty of Computer Science, Technion, 32000 Haifa, Israel, e-mail fredy@techsl.bitnet.

ISCIA 5, Fifth Int'l Symp. on Computer and Information Sciences: Oct. 30-Nov. 2, 1990, Cappadocia, Nevsehir, Turkey. Sponsors: Istanbul Technical Univ. et al. Submit full paper by **June 1, 1990**, to A. Emre Harmanci, Istanbul Technical Univ., Bilgi Islem Merkezi,

AYazaga, 80626 Istanbul, Turkey, phone 090 (1) 176-3254, fax 090 (1) 176-1734, e-mail harmanci@tritu.bitnet.

Int'l Workshop on Parallel Problem Solving from Nature: Oct. 1-3, 1990, Dortmund, West Germany. Sponsors: Parsytec GmbH et al. Submit four copies of extended abstract by **June 1, 1990**, to H. Muehlenbein, GMD-Z1, PO Box 1240, D-5205 St. Augustin 1, Federal Republic of Germany, phone 49 (2241) 142-405, fax 49 (2241) 142-889, e-mail grzia@dbngmd21.bitnet.

Micro 23, 23rd Symp. and Workshop on Microprogramming and Microarchitecture: Nov. 27-29, 1990, Orlando, Fla. Cosponsor: ACM. Submit five copies of full paper by **June 15, 1990**, to Chris Papachristou, Case Western Reserve Univ., Computer Engineering and Science Dept., Cleveland, OH 44106, phone (216) 368-5277, e-mail cap@alpha.cse.cwru.edu; or Vicki Allan, Utah State

Call for papers and referees for Computer

Computer seeks articles for inclusion in upcoming special issues.

Computer-Based Medical Systems has been selected as the theme for the **March 1991** edition. The issue will be devoted to examining the driving forces in the field, as well as historical directives and predictions of future directives. Manuscripts of a tutorial, descriptive, case-study, applications-oriented or pedagogic nature are immediately sought. Preference will be given to articles that delineate specific applications. For complete information, see the April issue of *Computer* (pp. 126-127).

Submittals and questions should be directed to John M. Long, Surgery and Computer Science, Univ. of Minnesota, 2701 University Ave. SE, Suite 201, Minneapolis, MN 55414, phone (612) 627-4850, fax (612) 625-3206, Bitnet hvm6006@umnacc; or Timothy J. Kriewall, Sarns/3M Health Care Group, 6200 Jackson Rd., Ann Arbor, MI 48103, phone (313) 971-3517, ext. 276, fax (313) 663-5062.

Abstracts are due by **June 1, 1990**, and eight copies of the full manuscript must be submitted by **July 1, 1990**. Authors will be notified of acceptance by **September 15, 1990**. The final version of the manuscript is due by **December 1, 1990**.

If you are willing to serve as a referee for this special issue,

For submittal to Computer, manuscripts must not have been previously published or currently submitted for publication elsewhere. Each manuscript should be no more than 32 typewritten, double-spaced pages long, including all figures and references. Each submittal should have a cover page that contains the title of the article, the full name(s) and affiliation(s) of the author(s), complete postal and electronic address(es), telephone number(s), a 300-word abstract, and a list of keywords identifying the central issues of the manuscript's contents. The final manuscript should have approximately 7,500 words and no more than 12 references.

send a note with a list of your technical interests and qualifications to Bruce D. Shriver, Editor-in-Chief, *Computer*, the University of Southwestern Louisiana, PO Drawer 42730, Lafayette, LA 70504-2730, phone (318) 231-5811, fax (318) 265-5472, Internet shriver@usl.edu

Computer Generated Music has been selected as the theme for the **July 1991** issue. The issue will be devoted to examining the driving forces in the field from a computational standpoint, assessing the limits of computer music in the general field of music, and discussing future desirable directions. See the April issue of *Computer* (p. 127) for complete information.

Abstracts are due by **August 30, 1990**, and four copies of the full manuscript and four audio cassettes are due by **October 30, 1990**. Notification of acceptance is set no later than **December 31, 1990**, and the final version of the manuscript must be submitted no later than **March 30, 1991**.

Submissions should be sent to Denis Baggi, Istituto Dalle Molle per Studi sull'Intelligenza Artificiale, Corso Elvezia 36, 6900 Lugano, Switzerland, phone 41 (91) 56 15 78, Europe e-mail denis@idsia.uu.ch; chx400.switch.ch, US e-mail baggi@berkeley.edu

If you are willing to review articles, please send a note to Denis Baggi or Bruce Shriver with a list of your technical interests and qualifications. Shriver's address is shown above.

Univ., Computer Science Dept., Logan, UT 84321-4205, phone (801) 750-2022, e-mail allan@usu.bitnet.

SIGGRAPH 90 Workshops, 17th Conf. on Computer Graphics and Interactive Techniques: Aug. 5-10, 1990, Dallas. Sponsor: ACM. Submit position paper by **June 15, 1990**, to Christine Barton, Morgan Guaranty Trust, 23 Wall St., New York, NY 10015, phone (212) 648-2355.

Fifth Int'l Conf. on Modeling Techniques and Tools for Computer Performance Evaluation: Feb. 13-15, 1991, Torino, Italy. Submit five copies of paper by **June 15, 1990**, to Gianfranco Balbo, Dip. di Informatica, Univ. di Torino, Corso Svizzera, 185, 10149 Torino, Italy, phone 39 (11) 771-2002, fax 39 (11) 751-603, e-mail balbo@itoin fo.bitnet.

19th AIRP Workshop on Applied Imagery Pattern Recognition: Oct. 17-19, 1990, McLean, Va. Sponsors: Society of Photographic Instrumentation Engineers, Rome Air Development Center. Submit abstract by **June 15, 1990**, to Brian Mitchell, ERIM, PO Box 8618, Ann Arbor, MI 48106, phone (313) 994-1200, ext. 2713.

EuroForum 90, Fourth European CFIS Forum: Oct. 11-12, 1990, Daresbury, Cheshire, UK. Submit abstract by **June 18, 1990**, to Kate Faulkner, EuroForum 90, ICL, Manchester M12 5DR, UK phone 44 (61) 223-1301, fax 44 (61) 223-1207.

AIDA 90, Sixth Conf. on Artificial Intelligence and Ada: Nov. 15-16, 1990, Reston, Va. Sponsors: George Mason Univ., et al. Submit four copies of complete paper by **June 29, 1990**, to AIDA 90, CS Dept., George Mason Univ., 4400 University Dr., Fairfax, VA 22030, phone (703) 323-2713, fax (703) 323-2630, e-mail aida@gmuva.gmu.edu.

1990 IEEE Workshop on Languages and Architectures for Automation: Dec. 19-21, 1990, Honolulu, Hawaii. Sponsors: Pacific Int'l Center for High Technology Research et al. Submit five copies of paper by **June 30, 1990**, to D.Y.Y. Yun, Univ. of Hawaii, 711 Kapiolani Blvd., Suite 200, Honolulu, HI 96813-5249, phone (808) 539-1532, fax (808) 941-1399.

IAPR Workshop on Machine Vision Applications: Nov. 28-30, 1990, Tokyo. Sponsor: Int'l Assoc. for Pattern Recognition. Submit four copies of summary by **June 30, 1990**, to Mikio Takagi, Inst. of Industrial Science, Univ. of Tokyo, 7-22-1 Roppongi, Minatoku, Tokyo 106, Japan, phone 81 (3) 479-0289, fax 81 (3) 423-2834, e-mail takagi@itk.iis.u-tokyo.ac.jp.

IFIP Working Conf. on Modeling in Computer Graphics: Apr. 8-12, 1991, Tokyo. Sponsor: IFIP TC 5/WG 5.10. Submit five copies of paper by **June 30, 1990**, to Toshiyasu I. Kunii, Information Science Dept., Faculty of Tokyo, Univ. of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan, phone 81 (3) 816-1783, fax 81 (3) 818-4607, e-mail b39756@tan sei.cc.u-tokyo.ac.jp.

Seventh Int'l Conf. on Data Engineering: Apr. 8-12, 1991, Kobe, Japan. Submit five copies of paper by **July 1, 1990**, to Nick J. Cercone, Center for Systems Science, Simon Fraser Univ., Burnaby, B.C., Canada, V5A 1S6, (604) 291-3229, e-mail Nick@CS.SFU.CA.

IEEE Trans. Computers plans a special issue on protocol engineering. Submit six copies of manuscript by **July 1, 1990**, to Ming T. (Mike) Liu, CIS Dept., Ohio State Univ., 2036 Neil Ave., Columbus, OH 43210-1277, phone (614) 292-1837, e-mail Liu@CIS.Ohio-State.Edu.

Third IEEE Conf. on Telecommunications: Mar. 17-20, 1991, Edinburgh, Scotland. Sponsor: Inst. of Electrical Engineers. Submit summary by **July 2, 1990**, to Conf. Services, IEEE, Savoy Place, London WC2R 0BL, UK, phone 44 (01) 240-1871, fax 44 (01) 240-7735.

10th IEEE Int'l Phoenix Conf. on Computers and Communications: Mar. 27-30, 1991, Scottsdale, Ariz. Sponsors: IEEE, IEEE Communications Society. Submit five copies of abstract and complete paper by **July 14, 1990**, to James A. Weeldreyer, Honeywell, Industrial Automation Systems Div., MS 2E5, 16404 N. Black Canyon Hwy., Phoenix, AZ 85023, phone (602) 863-5983, e-mail jw-ipcpc@enuxha.eas.asu.edu.

Fourth CSI/IEEE Int'l Symp. on VLSI Design: Jan. 5-8, 1991, New Delhi. Sponsors: Computer Society of India et al. Submit six copies of paper by **July 15, 1990**, to Lalit M. Patnaik, Computer Science and Automation, Indian Inst. of Science, Bangalore 560012, India, phone 91 (0812) 342-451, e-mail lalitm@shakti.turing.lalit@shakti.uu.net; or Adit D. Singh, Electrical and Computer Engineering Dept., Univ. of Massachusetts, Amherst, MA 01003, phone (413) 545-0188, e-mail singh@ecs.umass.edu.

Third UNB Artificial Intelligence Workshop: Oct. 1, 1990, Fredericton, N.B., Canada. Sponsor: Univ. of New Brunswick. Submit four copies of extended abstract by **July 15, 1990**, to B.G. Nickerson, School of Computer Science, Univ. of New Brunswick, PO Box 4400, Fredericton, N.B., Canada E3B 5A3, phone (506) 453-4566, fax (506) 453-3566, e-mail BGN@UNB.CA.

28th Allerton Conf. on Communication, Control, and Computing: Oct. 3-5, 1990, Monticello, Ill. Submit extended abstract for regular paper and summary for short paper by **July 16, 1990**, to Allerton Conf., c/o Donna J. Brown, Univ. of Illinois at Urbana-Champaign, Coordinated Science Lab, 1101 W. Springfield, Ave., Urbana, IL 61801, phone (217) 244-0581, e-mail djb@uicil.cs.uiuc.edu.

Electrosoft plans a special issue on software for electrical engineering education. Publisher: Computational Mechanics Publications. Submit paper by **July 22, 1990**, to D.J. Glover, Electrical and Computer Engineering Dept., 409 Dana Bldg., Northeastern Univ., Boston, MA 02115, phone (617) 437-3007.

Int'l J. Computer-Aided VLSI Design plans a special early-1991 issue on VLSI testing. Publisher: Ablex. Submit five copies of complete manuscript by **July 31, 1990**, to Sunil R. Das, Electrical Engineering Dept., Faculty of Engineering, Univ. of Ottawa, Ottawa, Ont., Canada K1N 6N5, phone (613) 564-3374, fax (613) 564-7681, e-mail das@uotw01 or Bit-net srdbp@uottawa.

Workshop on Parallel and Distributed Simulation: Jan. 21-23, 1991, Anaheim, Calif. Submit six copies of full paper by **July 31, 1990**, to Richard M. Fujimoto, School of Information and Computer Science, Georgia Inst. of Technology, Atlanta, GA 30332, phone (404) 853-9384, e-mail fujimoto@prism.ga tech.edu.

IEEE Software plans a special issue in March 1991 on testing and debugging. The issue will review the status of the two areas and present state-of-the-art techniques. Submit eight copies of article by **Aug. 15, 1990**, to Carl K. Chang, EECES Dept., Univ. of Illinois, Box 4348, Chicago, IL 60680, phone (312) 996-4860, fax (312) 413-1386, Compmail-c.k.chang, e-mail ckchang@uicbert.ee cs.uic.edu.

ETC 91, 1991 European Test Conf.: Apr. 17-19, 1991, Munich, West Germany. Sponsor: VDE (Zentralstelle Tagungen und Seminare). Submit four copies of abstract or full paper by **Aug. 31, 1990**, to ETC 91, c/o Bennetts Associates, Burnby Farm, Burridge, Southampton SO3 7YU, UK, fax (44) 489-579519.

CAIA 91, Seventh IEEE Conf. on Artificial Intelligence Applications: Feb. 24-28, 1991, Miami Beach, Fla. Submit paper by **Aug. 31, 1990**, to Tim Finin, Center for Advanced Information Technology, Unisys, 70E Swedesford Rd., PO Box 517, Paoli, PA 19301, phone (215) 648-2840.

ICSE 13, 13th Int'l Conf. on Software Engineering: May 13-16, 1991, Austin, Texas. Cosponsor: ACM. Submit eight copies of paper by **Sept. 14, 1990**, to David Barstow, Schlumberger Lab for Computer Science, PO Box 200015, Austin, TX 78720-0015.

First IEEE Int'l Workshop on Interoperability in Multidatabase Systems: Apr. 8-9, 1991, Kyoto, Japan. Submit seven copies of extended abstract by **Sept. 15, 1990**, to Marek Rusinkiewicz, Univ. of Houston, Computer Science Dept., Houston, TX 77204-3475, phone (713) 749-4791, e-mail marek@cs.uh.edu; or Yahiko Kambayashi, Kyushu Univ., Computer Science and Computer Engineering Dept., Hakozaki, Fukuoka 812, Japan, fax 81 (92) 641-1825, e-mail yahiko@csce.kyushu-u.ac.jp.

Fourth Int'l Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: June 2-5, 1991, Kauai, Hawaii. Sponsors: ACM et al. Submit four copies of extended abstract by **Oct. 1, 1990**, to Jim Bezdek, Computer Science Div., Univ. of West Florida, Pensacola, FL 32514, phone (904) 474-2784, fax (904) 474-2096, e-mail jbezdek@uwf.bitnet.

CAREER OPPORTUNITIES

RATES: \$12.00 per line, (ten lines minimum). Average five typeset words per line, eight lines per column inch. Add \$10 for box number. Send copy at least one month prior to publication date to: Marian B. Tibayan, Classified Advertising, COMPUTER Magazine, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380; fax (714) 821-4010.

In order to conform to the Age Discrimination in Employment Act and to discourage age discrimination, **COMPUTER** may reject any advertisement containing any of these phrases or similar ones: "...recent college grads...", "...1-4 years maximum experience...", "...up to 5 years experience...", "...10 years maximum experience..." **COMPUTER** reserves the right to append to any advertisement, without specific notice to the advertiser, "Experience ranges are suggested minimum requirements, not maximums." **COMPUTER** assumes that, since advertisers have been notified of this policy in advance, they agree that any experience requirements, whether stated as ranges or otherwise, will be construed by the reader as minimum requirements only.

DARTMOUTH COLLEGE

The Thayer School of Engineering at Dartmouth College invites applications for tenure-track appointments. Of special interest are candidates with experience in analog or digital VLSI circuits, but outstanding candidates in all areas of computer engineering are encouraged to apply. Current research in computer engineering at Dartmouth has focused on the design of special-purpose computational structures. This effort is supported by the Thayer Rapid Prototyping Facility, a laboratory under construction for the design, implementation, test, and evaluation of experimental digital systems. Candidates must be able to teach effectively at both the undergraduate and graduate levels and develop a strong research program within an interdisciplinary engineering environment.

Dartmouth offers excellent computing facilities and a new, 1000 sq. ft., Class 100/1000 clean room facility for device processing.

Interested persons should submit a resume and names of three references to Prof. Charles Hitchcock, Thayer School of Engineering, Dartmouth College, Hanover, New Hampshire 03755. Dartmouth is an equal opportunity, Affirmative Action employer and ENCOURAGES APPLICATIONS FROM WOMEN AND MEMBERS OF MINORITY GROUPS.

THE GEORGE WASHINGTON UNIVERSITY Endowed Professorships School of Engineering and Applied Science

The School of Engineering and Applied Science of The George Washington University is seeking to fill two newly endowed professorships. The L. Stanley Crane Professorship will be appointed in the Department of Electrical Engineering and Computer Science and the A. James Clark Professorship in the Department of Civil, Mechanical and Environmental Engineering. One professorship will be appointed in the 1990-91 academic year and the other in the 1991-92 academic year.

Each appointment will be coordinated with programmatic developments for a new campus that the University is opening in Northern Virginia in 1991. Initially, academic programs on that campus will focus primarily on graduate education and research.

Individuals with a preeminent record of accomplishments are sought; and academic background is desirable. The successful candidate for each professorship will have an outstanding record of research and publications, a sustained record of performance within industrial and/or government spheres, and recognition in both domestic and international organizations.

A review of resumes and supporting documentation will commence on April 16, 1990 and continue until the positions are filled. Submit credentials to:

Professor Sam Rothman, Chair
Selection Committee for Endowed
Professorships
School of Engineering and Applied
Science

The George Washington University
Washington, D.C. 20052

The George Washington University is an Equal Opportunity/Affirmative Action Employer.

THE ASIAN INSTITUTE OF TECHNOLOGY P.O. Box 2754, Bangkok 10501, Thailand

The Asian Institute of Technology has openings in the Division of Computer Science for faculty positions in Operating Systems and Software Engineering. Candidates should have a Ph.D. in Computer Science or closely related fields. Duties include teaching and supervising graduate students curriculum development and professional activities. Rank and salary commensurate with experience and qualification. Selection will start in September for appointment to be effective in January 1991. Applications should reach AIT (Vice President for Academic Affairs or Chairman, Division of Computer Science) by August 15, 1990.

TRINITY COLLEGE

The Department of Engineering and Computer Science announces a tenure-track position, pending approval, in the field of Mechanical Engineering. It, therefore, invites applications from outstanding candidates for a position at the Assistant or Associate Professor-level commencing September, 1990, in the areas of THERMODYNAMICS/HEAT TRANSFER or ROBOTICS/CONTROLS. Experimental background highly desirable. The position involves graduate and undergraduate instruction and research, and a doctoral degree is a prerequisite. We are interested in receiving applications from women and minorities. Trinity College is an equal opportunity/affirmative action employer. Please send resume to Professor Joseph D. Bronzino, Chairman, ECS Department, Trinity College, Hartford, CT 06106. Consideration of applications will begin immediately and the search will remain open until the position is filled.

WEST VIRGINIA UNIVERSITY

The Department of Electrical and Computer Engineering at West Virginia University seeks candidates for tenure-track faculty positions with research specializations in all areas of electrical and computer engineering. Special emphasis at this time for candidates is in Computer Engineering. Candidates should possess strong teaching qualifications as well.

The Department of Electrical and Computer Engineering offers degree programs at the bachelor's, master's and doctoral levels. Departmental facilities include numerous workstations, several research and instructional laboratories. The University supports computing with a network of IBMs and VAXes.

Morgantown is a very pleasant place in which to live and work while the cost of living is rather low. The city has many cultural and sporting activities available. Morgantown is located about one and a half hours south of Pittsburgh, PA. In addition, West Virginia was recently rated as having the lowest crime rate in the nation.

Rank and salary will depend on qualifications. Applicants should send a cover letter stating specific area of specialization and a curriculum vitae with names and addresses of at least three references to: Chairman, Department of Electrical and Computer Engineering, West Virginia University, P.O. Box 6101, Morgantown, WV 26506-6101.

Women, minorities, and members of other protected groups are strongly encouraged to apply. West Virginia University is an Equal Opportunity/Affirmative Action Employer.

The Department of Computer Science is seeking qualified candidates for a sabbatical leave replacement position beginning in September 1990. Applicants should have a Ph.D. in computer science (ABD will be considered) and should have an interest in teaching and research at a selective liberal arts college with a CSAB accredited computer science program. The area of specialization is open.

Rank and salary are competitive and commensurate with background and experience. There is a possibility that the position may be extended to a second year.

Applicants should submit resume, graduate transcript, and three letters of recommendation to Dr. Benjamin D. Haytock, Department of Computer Science, Allegheny College, Meadville, PA 16335. We will begin reviewing applications on May 15. However applications will be considered until the position has been filled or a decision has been made not to fill it. Allegheny College is an equal opportunity employer.

NAVAL POSTGRADUATE SCHOOL

The Computer Science Department invites applications for faculty positions at all levels. Our primary interests are in the areas of operating systems and programming languages. Our secondary interests are in the areas of visual data processing, graphics, and computer architecture (especially real-time and parallel-processing aspects of the three). Applicants should have a Ph.D. in Computer Science or a closely related field and be committed to high-quality teaching and research. Senior applicants must have distinguished research records. Appointments can begin at any time.

The Department offers M.S. and Ph.D. degrees in computer science, but no undergraduate degrees. Currently, 110 students are enrolled in the M.S. program and five in the Ph.D. program. Students are military officers or civilian employees of the Department of Defense and are fully supported by their sponsoring organization during their studies. Departmental facilities (supported by eight full-time computer professionals) include six instructional and research laboratories with extensive state-of-the-art equipment. Faculty normally teach for two quarters and perform research for two quarters per year. The Monterey-Carmel area provides a pleasant coastal climate and easy access to Silicon Valley companies.

Send a detailed resume, an abstract of your best recent research, and letters of reference to:

Faculty Search Committee
Computer Science Department, Code 52
Naval Postgraduate School
Monterey, CA 93943
Telephone (408) 646-2449
An Equal Opportunity/Affirmative Action Employer

Computer Engineer for NE Ohio Biomedical Instruments Sensors manufacturer to design & implement microprocessor based computer systems for monitoring electrochemical sensors in a biomedical product, using Intel 80376 microprocessor, Intel 82370 for Direct Memory Access Channels (DMA) Control & Interrupt Control, Advanced-Micro Device's (AMD's) Tachip for interfacing Central Processing Unit with sensors & AMD's programmable logic devices; develop a real time concurrent operating system using Intel's RMK real time kernel to control several sensors simultaneously; design ROM resident initialization code; design system software in C & Assembly Languages to drive the sensors, to acquire the data from the sensors, & perform mathematical analysis on the data; design code for displaying results on screen in real time. No exp. required in above duties but applicants will qualify with a B.S.-Engineering with a major in Computer Engineering & coursework including at least the following: 3 classes in Computer Design (with lab for at least 2 of the classes), 2 classes in Electronic Circuits, 2 classes in Data Structures, 2 classes in Numerical Analysis, & 1 class each in Computer Architecture, Operating Systems, Micro-processors, Software Engineering, & Statistics. Must have proof of legal authority to work permanently in U.S. 40 hrs./wk. M-F 8AM-5PM. \$31,540/yr. Send resume in duplicate (NO CALLS) to L. Eggleston, JO#1084528, Ohio Bureau of Employment Services, P.O. Box 1618, Columbus, OH 43216.

UNIVERSITY OF SOUTH FLORIDA

Chairperson Department of Computer Science and Engineering

The Department of Computer Science and Engineering at the University of South Florida invites nominations and applications for the position of Chairperson. The Department offers Bachelor's degrees in Computer Science (accredited by CSAB) and in Computer Engineering (accredited by ABET), Master's and Ph.D. degree programs. The current faculty size is 15, with several new positions to be added over the next two years. The four broad areas of research emphasis chosen by the faculty are

- Computer Architecture/VLSI Design & Test,
- Computer Vision/Graphics/Image Processing,
- Artificial Intelligence/Expert Systems, and
- Software Engineering.

The Department is relatively young and ambitious, with a rapidly expanding research program. An experienced individual is sought as Chairperson to lead the Department to a position of national/international recognition.

The Department shares a new 12 million dollar building with the Department of Electrical Engineering. The Department research

network includes a substantial number of SUNs and VAXes, an INTEL 2/386 hypercube, a variety of specialized graphics and image processing equipment, and a number of other resources. Additional computing resources are available on the College computing network and the University network.

The University of South Florida is the second largest university in the State of Florida, and the forty-second largest in the nation, with an enrollment of well over 30,000. USF occupies a 1,700-acre campus in the city of Tampa, one of the largest and fastest growing metropolitan areas in Florida, offering a wide variety of cultural, entertainment, sports and outdoor activities. The quality of life is excellent and the cost of living is moderate. The area near the University is experiencing dramatic growth in high-technology industry and medical facilities. The faculty of the Department have interactions with a number of companies in the area, including AT&T, E-Systems, GTE Data Services, Harris, Hercules, Honeywell and IBM. A PBS public-TV channel is located on campus: other television links tie the University to industry and remote campuses.

Applicants should send a resume and the names of three references to the Department Chairperson Search Committee, Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620.

The University of South Florida is an equal opportunity and affirmative action employer.

UNIVERSITY OF VERMONT Visiting Faculty Positions in Computer Science

The Department of Computer Science and Electrical Engineering invites applications for up to two visiting faculty openings in Computer Science, commencing September, 1990. A doctorate in Computer Science, or the equivalent is required. The level of assistant or associate professor will be dependent upon the candidates' qualifications. Responsibilities, in addition to research, include teaching in mainstream computer science at both the undergraduate and graduate level in two or more of the following areas: operating systems, programming languages, database systems, computational complexity and algorithm theory, theory of computation, as well as the presentation of an advanced graduate course or seminar. Applications will be accepted until the positions are filled. Please submit a resume, including teaching experience, a publication list, and the names, addresses, and phone numbers of three references to: Dr. Kenneth Golden, Chairperson; University of Vermont; Department of Computer Science and Electrical Engineering; Votey Building, Burlington, VT 05405, (802) 656-3330. Internet & CSnet: csearch@uvm.edu; USENET: unetvum-gen@uvm.edu; BITNET: csearch@uvm-gen.uvm.edu. The University of Vermont is an Affirmative Action Equal Opportunity Employer and encourages applications from women and members of minority groups.

UNIVERSITY OF WEST FLORIDA
Division of Computer Science

Applicants are invited for two tenure track positions in the Division of Computer Science. The successful candidates must hold a Ph.D. in Computer Science or closely related field, and have a depth of knowledge in one or more of the following areas: constructivist approaches to AI and cognitive science, expert systems, knowledge acquisition, neural nets, and image processing.

The Division of Computer Science is an independent academic unit reporting to the Vice President of Academic Affairs. It offers a B.S. and M.S. degree in computer science and over the next few years will continue the development of its research and graduate programs including doctoral level work. Extensive opportunities exist for local research involvement with the military and the medical communities. The Division currently enrolls about 650 undergraduate and 200 graduate students. In addition, the Division houses the Institute for the Interdisciplinary Study of Human and Machine Cognition. The research environment includes a Solbourne Series 4, Sun SPARCstations, Lisp machines, IBM mainframe, and numerous Macintosh computers.

Competitive salaries combined with a very low cost of living enhance the excellent lifestyle available in northwest Florida.

Vitae and letter of application should be sent to Dr. Theodore Elbert, Division Head, Computer Science, The University of West Florida, Pensacola, FL 32514. The positions will remain open until filled. UWF is an EO/AA institution.

THE GEORGE WASHINGTON
UNIVERSITY
Faculty Positions
School of Engineering and
Applied Science

New and challenging opportunities for faculty appointments, both tenure-track and research, at the ranks of assistant, associate, full, and distinguished professor are available in the School of Engineering and Applied Science beginning Fall Semester 1990 and Spring Semester 1991 in various specialty areas. These positions may require faculty to participate in new programs in Northern Virginia and other off-campus locations in the Washington metropolitan area.

The George Washington University is located in the center of Washington, D.C. This metropolitan area sustains the second largest concentration of research and development activity in the United States, creating a continuing demand for rigorously trained engineers and many research opportunities. The School of Engineering and Applied Science is organized into four academic departments: the Department of Civil, Mechanical and Environmental Engineering; the Department of Electrical Engineering and Computer Science; the Department of Engineering Administration; and the Department of Operations Research.

Faculty will be expected to teach undergraduate and/or graduate courses, to interact with faculty colleagues in interdisciplinary specialty areas, and to conduct and promote sponsored research in their specialty areas. Two senior faculty members selected will be assigned additional responsibilities to the above in performing the duties of a Research Director and Chief Scientist.

Candidates with expertise in the following specialty areas are especially valued: aerospace, aerospace mission analysis, artificial intelligence, communications, composites, computational fluids and solids, computer-integrated manufacturing, computer science, expert systems, foundational structural engineering, human factors engineering, marketing of technology, mechanics of materials and structures, reliability, robotics, software engineering, structures, systems analysis and engineering, systems engineering and management (e.g., energy and environmental management, project and program management, biotechnology management, total quality management), and VLSI. Candidates in other engineering or applied science specialty areas are also encouraged to apply.

Candidates should have an earned doctorate in engineering or a related discipline and research experience with an interest in both teaching and research. Good communication skills, both oral and written, are essential. Registration as a professional engineer is desirable. Salary and rank will be commensurate with qualifications and experience. Applications will be reviewed beginning April 16 and will be accepted until the positions are filled. Applicants should send their vitae including a list of publications with three references to:

Office of the Dean
School of Engineering and Applied
Science
The George Washington University
Washington, D.C. 20052
The George Washington University is an
Affirmative Action/Equal Opportunity
Employer.

UNIVERSITY OF MISSOURI-
COLUMBIA

The Department of Electrical and Computer Engineering invites applications for tenure-track positions at the assistant, associate or full professor level in the areas of quantum well materials and devices, optoelectronics or optical signal processing and telecommunications. Responsibilities include teaching undergraduate and graduate courses, student advising, and developing and conducting sponsored research programs. Candidates must have an earned doctorate in Electrical and Computer Engineering or a related discipline, and the potential for, and commitment to, developing sponsored research. The department currently has strong research programs in computer vision/pattern recognition, semiconductor materials and devices including

VLSI and power electronics. Interested applicants should send a resume, a description of teaching and research interests and a list of three references to: Jon Meese, Chairman, Department of Electrical and Computer Engineering, University of Missouri-Columbia, Columbia, Missouri 65211. Immigration status of non-United States citizens should be stated. Affirmative Action/Equal Opportunity Employer.

UNIVERSITY OF SOUTH FLORIDA
Faculty
Department of Computer Science
and Engineering

The Department of Computer Science and Engineering is seeking applications for several tenure track faculty positions starting in the 1990-91 academic year. The Department offers Bachelor's degrees in Computer Science (accredited by CSAB) and in Computer Engineering (accredited by ABET), Master's and Ph.D. degree programs. The current faculty size is 15. Our Department's areas of research emphasis include

- Computer Architecture/VLSI Design & Test
- Computer Vision/Graphics/Image Processing
- Artificial Intelligence/Expert Systems, and
- Software Engineering

The Department shares a new 12 million dollar building with the Department of Electrical Engineering. The Department research network includes a substantial number of SUNs and VAXes, and INTEL 2/386 hypercube, a variety of specialized graphics and image processing equipment, and a number of resources. Additional computing resources are available on the College computing network and the University network. The department faculty are extremely aggressive in research activities supported by Federal, State and industrial agencies.

The University of South Florida is the second largest university in Florida, with a current enrollment of over 30,000. It is located on a 1,700-acre campus in the city of Tampa, one of the largest and fastest growing metropolitan areas in Florida, offering a wide variety of cultural, entertainment, sports and outdoor activities. The quality of life is excellent and the cost of living is moderate. The area near the University is experiencing dramatic growth in high-technology industry and medical facilities. The faculty of the Department have interactions with a number of companies in the area, including AT&T, E-Systems, GTE Data Services, Harris, Hercules, Honeywell and IBM. A PBS public-TV channel is located on campus; other television links tie the University to industry and remote campuses.

Applicants should send a resume and the names of three references to Faculty Search Committee, Computer Science and Engineering, University of South Florida, Tampa, FL 33620.

The University of South Florida is an equal opportunity and affirmative action employer.

**COMPUTER SCIENCE
AND TECHNOLOGY RESEARCH
Program Officer**

The NATIONAL RESEARCH COUNCIL is seeking a Program Officer for its Computer Science and Technology Board. Responsibilities include designing projects, identifying national experts for project committees, conducting background research, providing analytical support for ongoing projects, maintaining liaison with the computer science community and drafting project documents. An advanced degree in computer science, electrical engineering, public policy or a related field and at least 4 years computer-related technology assessment, applications, and development—including research or management—experience is desired. Ability to communicate technical concepts to a non-technical audience and familiarity with science and technology policy issues also required. We offer a salary commensurate with experience and an exceptional benefits package. Please send resume in confidence to: NRC/CSTB, HA 560 (MK), 2101 Constitution Avenue, N.W., Washington, D.C. 20418. EOE.

**TRINITY COLLEGE
Computer Science Faculty**

Trinity College is establishing a Department of Computer Science and is seeking candidates at any level for a tenure track position starting in August 1990. The successful candidate will join two other computer science faculty in the continued development of the computer science major which was begun in 1985 and is presently offered through the Department of Engineering and Computer Science.

Candidates must have a Computer Science Ph.D., a strong commitment to excellence in undergraduate teaching, a willingness and ability to participate in the continued development of a strong liberal arts computer science major and the potential to pursue an active research program. Applicants in all areas of computer science will be considered.

Trinity College is a selective liberal arts college with a strong commitment to the sciences. In addition to computer science, the College offers majors in engineering, mathematics, chemistry, biochemistry, physics, biology and psychology. The College's academic computing facilities include a VAX 8350, a network of Sun 3/50 and 3/60 workstations and numerous personal computers. Trinity College is an equal opportunity/affirmative action employer, and has a primary goal of increasing the number of women and minority faculty in the sciences. Please send application letter, vita and letters of reference to Professor Ralph Walde, Department of Engineering and Computer Science, Trinity College, Hartford, CT 06106. Consideration of applications will begin immediately and the search will remain open until the position is filled.

COMPUTER PROGRAMMER ANALYST

Min. 2 yr. exp. with bachelor's degree field of science to write programs relating to various Medical tests conducted for our clients, including types of test, fee involved, results of test. Will also convert medical test problems to detailed logical flow charts for coding into computer language and solution through tests. Salary \$39,900.00/year. Send resume to: Central Reference Laboratory, 1621 So. Sinclair Street, Anaheim, CA 92806. Attn: Mr. Dalvanian.

FACULTY POSITION AVAILABLE

Applications are invited for a tenure-track faculty position at the assistant professor level. The duties of the job are to teach undergraduate and graduate courses in computer science and engineering, to serve on departmental committees, to direct graduate student thesis and dissertations, and to perform scholarly research. The applicant must be qualified to teach and perform independent scholarly research in the subject areas of fault-tolerant and ultra-reliable computing, computer networks, distributed computing, and real-time systems. These qualifications must be evidenced by the applicant having taken or taught courses in one or more of these areas and by having had research experience that led to publication of scholarly papers dealing with one or more of the subjects. A Ph.D. in computer science and engineering is required. The salary is \$4633.33 per month. Apply at the Texas Employment Commission, Austin, Texas or send resumes to the Texas Employment Commission, TEC Building, Austin, Texas 78778, J.O. #5515102. Ad Paid by An Equal Employment Opportunity Employer.

COMPUTER CONSULTANT

Computer Consultant for Southwest Ohio International Computer Consultant Group to be responsible for the development of application systems for clients including its design and analytical phase; responsible for writing all Documents for Requirement/Functional/Implementation specifications; perform the entire development of the system on distributed workstations running UNIX operating systems and using Object oriented language C+++. No exp. required in above duties but applicants will qualify with a MS degree in Computer & Information Science or Computer Science, and at least two yrs. systems analysis exp. in the area of applications exp. and retail systems. 40 hrs/wk., 8AM-5PM. \$834.00/wk. Must have proof of legal authority to work permanently in U.S. Qualified applicants send resume in duplicate (NO CALLS) to J. Davies, JO#1232904, Ohio Bureau of Employment Services, P.O. Box 1618, Columbus, OH 43216.

**THE GEORGE WASHINGTON
UNIVERSITY
Marketing of Technology
Tenure-Track Faculty Position**

A tenure-track faculty position is available in the graduate program of the Department of Engineering Administration, School of Engineering and Applied Science, starting Fall Semester 1990. The position requires:

- Ability to develop and teach courses pertinent to: marketing of technology, and/or technology assessment, and/or technology transfer.
- Ability to generate and conduct sponsored research supported by funds from sources outside the University.
- Ability to academically advise degree candidates in fields of Engineering Administration at the masters, professional, and doctoral level.

The George Washington University is located in the center of Washington, D.C. This metropolitan area sustains the second largest concentration of research and development activity in the United States, creating a continuing demand for rigorously trained engineers and many research opportunities. The Department of Engineering Administration conducts a major off-campus degree program at locations in the Washington metropolitan area and the person chosen for this position will participate in that program.

The successful candidate will have an earned doctorate in an engineering or applied science discipline and will have been professionally associated for at least five years with some aspect of technology management. Salary and entry level academic rank will depend on qualifications. Applications will be reviewed beginning May 15, 1990, and will be accepted until the position is filled. Send vita and three references to:

Professor Homer B. Sewell, Chairman
Department of Engineering Administration
School of Engineering and Applied Science
The George Washington University
Washington, D.C. 20052
Affirmative Action/Equal Opportunity
Employer

**HAWAII PACIFIC COLLEGE
Computer Science**

Hawaii Pacific College invites applications for computer science faculty, appointments to commence August 24, 1990. Applicants should hold the Ph.D. in computer science, or an appropriate master degree plus exceptional professional experience. The applicant should be able to demonstrate research and teaching potential. Positions will be filled at the Assistant or Associate Professor rank, depending on credentials and background. Salary is competitive and commensurate with experience.

Applicants should submit a curriculum vitae and three references to: Dr. Arnold Lipkind, Academic Dean, Hawaii Pacific College, 1188 Fort St., Suite 440, Honolulu, Hawaii 96813.

Hawaii Pacific College is an equal opportunity/affirmative action employer.

COMPUTER SCIENCE RESEARCHERS

The Centre de recherche informatique de Montreal (CRIM) is a non-profit corporation focusing on excellence in research and development and on the transfer of information technologies and their applications.

CRIM is seeking Computer Science Researchers to join its research groups.

RESPONSIBILITIES
Participation in joint university-industry research projects. Domain: software engineering. Specialization: automatic support and assistance for software design and reuse; reverse engineering. Knowledge of Artificial Intelligence is an asset.

QUALIFICATIONS

- The applicant must hold a Ph.D. in computer science with sufficient publication. Industrial experience is sought.

- Priority will be given to Canadian citizens and landed immigrants.

- Knowledge of French is an asset.

Interested candidates may apply by sending a resume before May 31, 1990 to:

Dr. Jacqueline Bourdeau
Centre de recherche informatique de
Montreal
1550 de Maisonneuve Blvd. West, Suite
1000
Montreal (Quebec) H3G 1N2

SOUTHERN METHODIST UNIVERSITY Department of Computer Science and Engineering

The Department of Computer Science and Engineering (CSE) invites applications for faculty positions at both senior and junior levels. Applicants for the senior position should have an outstanding reputation in academic pursuits and a strong publication record. Applicants for the junior positions should have a Ph.D. degree and demonstrated research and teaching experience in computer science. We are interested in applicants in the following areas: computer architecture, computer systems, computer languages, and database systems.

SMU is a private university with approximately 9,000 students. CSE is in the School of Engineering and Applied Science, where a close working relationship exists with the Department of Electrical Engineering. The Department has extensive contacts with computer-related and engineering-oriented industrial firms that distinguish Dallas as one of the top centers for high technology.

CSE presents a balanced program of research and education at all levels and has been offering Ph.D. degrees since 1970.

Applicants should send a complete resume, including the names of three references to: J.L. Kennington, Chair, Department of Computer Science and Engineering, Southern Methodist University, Dallas, Texas 75275-0122.

SMU is an equal opportunity/affirmative action employer. Applications from women and minorities are particularly encouraged.

NORWICH UNIVERSITY

Tenure Track position in the Department of Computer Science and Engineering starting July 1990. Rank and salary commensurate with experience. Ph.D. desired, but will consider Masters with significant experience. Recently unified Computer Science and Computer Engineering program. Resources include: VAX Cluster, MicroVax, AT&T 3B2 400, VAX Stations and numerous PCs. New faculty member will be responsible for teaching courses in an undergraduate Computer Science curriculum and assisting in curriculum and laboratory development. Expertise in any of the following areas desired: Computational Algorithms, Simulation, Operating Systems, Computer Graphics, Programming Languages, AI and Software Methodology. A strong commitment to undergraduate education is essential. Applications will be accepted until the position is filled. Norwich University is located in an area of Central Vermont that offers small town or rural living with outstanding recreational opportunities. Must be U.S. Citizen or Permanent Resident. Send resume and references to Dr. Michael C. Murphy, Head, Computer Science and Engineering Department, Norwich University, Northfield, Vermont 05663. EOE. Women and minorities encouraged to apply.

U S WEST CHAIR in TELECOMMUNICATIONS

University of Minnesota Department of Computer Science

The University of Minnesota invites nominations and applications for the U S WEST Land Grant Chair in Telecommunications. Of particular interest are candidates with a strong research background appropriate to software technology for broadband public communications and computing environments.

Candidates for the position must be capable of providing leadership in collaborative research with industry and contributing significantly to the current research programs at the University of Minnesota, which include network architecture and protocol design for broadband high-speed communications, voice-data-video integration, interconnection of local area networks, performance analysis and modelling and multimedia communications.

Applicants and nominees must have an outstanding research record, a strong interest in teaching, and a commitment to the development of a nationally recognized research program in telecommunications.

Interested persons should write or call Professor David Fox, Chair of the Search Committee for the U S WEST Chair, Department of Computer Science, University of Minnesota, 200 Union Street, Minneapolis, MN 55455. Tel 612-625-0726. Deadline for receipt of application has been extended to May 31, 1990.

The University of Minnesota is an equal opportunity educator and employer and specifically invites and encourages applications from women and minorities.

Books You Can't Put Down

The technical literature is ever expanding. Which books should you read? Which should you avoid?

IEEE Software reviews books with your needs in mind, whether you are a practitioner or educator. Each issue, Mike Lutz selects recent books spanning software's diverse theories, practices, and philosophies and has experts in the field evaluate them. Our Book Reviews department has an eclectic mix to keep you abreast.

We're what a technical magazine should be: Practical. Authoritative. Lucid. Direct.

For subscription information, write *IEEE Software*, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; call (714) 821-8380; or use the reader-service card.

IEEE Software

*The state of the art
about the state of the practice.*



IEEE COMPUTER SOCIETY

A member society of the Institute of Electrical and Electronics Engineers, Inc.

Executive Committee

President: Helen M. Wood*
National Oceanic and Atmospheric Administration
FB 4, Rm. 1069, Code E/SP
Washington, DC 20233
(301) 763-1564

President-Elect: Duncan H. Lawrie*
Past President: Kenneth R. Anderson*

VP, Conferences and Tutorials: Laurel V. Kaleda (1st VP)*
VP, Standards: Paul L. Borriell (2nd VP)*
VP, Area Activities: Gerald L. Engel*
VP, Education: Ronald G. Hoelzeman†
VP, Membership and Information: Barry W. Johnson†
VP, Press Activities: James H. Aylor†
VP, Publications: Sallie V. Sheppard†
VP, Technical Activities: Mario R. Barbacci*

Secretary: David Pessel*
Treasurer: Joseph Boykin†
Division V Director: Edward A. Parrish, Jr.*
Division VIII Director: Roy L. Russo*
Executive Director: T. Michael Elliott†
*voting member of the Board of Governors
†nominating member of the Board of Governors

Board of Governors

Term Expiring 1990:

Vishwani Agrawal, Mario R. Barbacci,
Ming T. (Mike) Liu, Yata N. Patt, Donald E. Thomas,
Benjamin W. Wah, Ronald Waxman

Term Expiring 1991:

P. Bruce Berra, Michael Evangelist,
Ted Lewis, Raymond E. Miller, Earl E. Swartzlander, Jr.,
Joseph E. Urban, Thomas W. Williams

Term Expiring 1992:

Alicia I. Ellis, Tadao Ishikawa,
David Pessel, Sallie V. Sheppard, Bruce D. Shriver,
Harold Stone, Wing N. Toy

Next Board Meeting

June 8, 1990, 8:30 a.m.
Le Meridien, San Francisco, CA

Senior Staff

Executive Director: T. Michael Elliott
Editor and Publisher: H. True Seaborn
Director, Computer Society Press: Eugene M. Falken
Director, Conferences and Tutorials: Anne Marie Kelly
Director, Finance: Tod S. Heisler
Director, Board and Administrative Services: Violet S. Doan

Computer Society Offices

Headquarters Office
1730 Massachusetts Ave. NW
Washington, DC 20036-1903
Phone (202) 371-0101
Telex: 7108250437 IEEE COMPSO
Fax: (202) 728-9614

Publications Office
10662 Los Vaqueros Cir.
PO Box 3014
Los Alamitos, CA 90720-1264
Membership and General Information: (714) 821-8380
Publication Orders: (800) 272-6657
Fax: (714) 821-4010

European Office
13, Ave. de L'Aquilon
B-1200 Brussels, Belgium
Phone: 32 (2) 770-21-98
Fax: 32 (2) 770-85-05

Asian Office
Ooshima Building
2-19-1 Minami-Aoyama, Minato-ku
Tokyo 107, Japan
Phone: 81 (3) 408-3118
Fax: 81 (3) 408-3553

Use the Reader Service Card to obtain information on:

- Membership application—student #203, others #202
- Periodicals subscription form for individuals #200
- Periodicals subscription form for organizations #199
- Publications catalog #201
- Standards working groups list #195
- Compmail+ international electronic mail/database brochure #194
- Technical committee list/application #197
- Chapters lists, start-up procedures—student/regular #193
- Student scholarship information #192
- Awards description/nomination forms #198
- Volunteer leaders/staff directory #196
- IEEE senior member application #204

To check membership status or report a change of address, call the IEEE toll-free number, 1-800-678-4333. Direct all other Computer Society related questions to the Publications Office.

Purpose

The IEEE Computer Society advances the theory and practice of computer science and engineering, promotes the exchange of technical information among 100,000 members worldwide, and provides a wide range of services to members and nonmembers.

Membership

Members receive the acclaimed monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others seriously interested in the computer field.

Publications and Activities

Computer. An authoritative, easy-to-read magazine containing tutorial and in-depth articles on topics across the computer field, plus news, conferences, calendar, interviews, and new products.

Periodicals. The society publishes six magazines and five research transactions. Refer to membership application or request information as noted above.

Conference Proceedings, Tutorial Texts, Standard Documents. The Computer Society Press publishes more than 100 titles every year.

Standards Working Groups. Over 100 of these groups produce IEEE standards used throughout the industrial world.

Technical Committees. More than 30 TCs publish newsletters, provide interaction with peers in specialty areas, and directly influence standards, conferences, and education.

Conferences/Education. The society holds about 100 conferences each year and sponsors many educational activities, including computing science accreditation.

Chapters. Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

European Office

Payments for Computer Society membership and publication orders are accepted by checks in Belgian, British, German, Swiss, or US currency. Checks in US funds must be drawn on a US bank. Payment may also be made by American Express, MasterCard, or Visa credit cards.

Asian Office

Payments for Computer Society membership and publication orders are accepted by checks in Japanese or US currency. Checks in US funds must be drawn on a US bank. Payment may also be made by electronic fund transfer to the Bank of Tokyo, Akasaka Branch, Toza acct. 0767956; the credit receiver is also the IEEE Computer Society Headquarters Office. Payment may also be made by American Express, MasterCard, or Visa credit cards.

Ombudsman

Members experiencing problems — magazine delivery, membership status, or unresolved complaints — may write to the ombudsman at the Publications Office.

BOOK REVIEWS

Editor: Guy Johnson, Department of Information Technology, Rochester Institute of Technology, 1 Lomb Memorial Drive, Rochester, NY 14623.

Silicon Dreams: Information, Man, and Machine

Robert W. Lucky (St. Martin's Press, New York, 1989, 411 pp., \$16.95)

As the subtitle indicates, this is a wide-ranging book. Lucky attempts to weave together those aspects of information theory common to computer-aided processing of text, sound, speech, and pictures. Nearly every page reflects his passion for information representation and coding and its application to current and future computer-based systems. However, his enthusiasm and its expression are two of the book's several shortcomings.

Lucky tries to say so much, to explore so many relations, and to ponder so many possibilities that the book wanders in several directions. And although he admits in the preface that trying to cover so much has yielded mixed results, Lucky does believe that everyone who reads the book will learn something. However, the frequent side comments and parenthetical expressions make for very frustrating and often tedious reading.

A recurrent theme is the replacement of people by computing machines in work and work-related social activities. Human mental facilities are repeatedly described in terms of computer hardware and software components. Lucky cautions against reading too much into his analogies between human mental activity and the computer execution of stored programs, but while reading the book I began to think he does believe there are subroutines in his mind. More importantly, Lucky left me with the feeling that humans are inferior to computers in some essential, but unspecified, sense. It is as if he really wanted to explore these ideas further, but for some reason did not. The relation of human mentality to computer intelligence clearly interests Lucky, but he doesn't deal with it directly. If he had, the book might have been more personal, more controversial, perhaps more illuminating, and even fun to read.

By far the most interesting and involving discussion explores information and language expressed as text. The detailed explanations of the frequencies of letter occurrences in English and of text-compression methods are well written, and in

spite of their technical nature the explanations would be accessible to interested readers. Many of the examples of coding and statistical text generation are fascinating. In fact, there are many surprising, informative, and stimulating ideas and facts throughout the book. But Lucky's attempt to link these discussions through the concepts of information and entropy does not work for several reasons.

First, the level of discussion often slips from common-sense examples and lan-

guage into very technical and arcane detail, interrupting the book's continuity and making one wonder who the intended audience is.

A book claiming to explore social aspects of information technology should better address the serious questions about progress and the effects of technology on modern life.

Second, the breadth of the book leads to anomalies in exposition. For example, at the end of Chapter 1, Lucky writes, "We cannot ignore the human factor in studying information, and we shall discuss it wherever possible . . ." However, in describing the theory of information in the next chapter, he states, "Semantic aspects of communication are irrelevant to the engineering problem." In a narrow, technical sense this latter statement might be true, but exploring the effects of technology on our social lives requires finding out how the engineering problems of communication are relevant to the meanings of messages. Lucky sets us up to expect analyses of the social effects

of information technology, but this is really a technical book.

Third, the relations among information theory, entropy, and human language are profound, but Lucky does not really get at these deep connections. The book makes it easy to see connections between the information theoretic entropy of a message and the redundancy of written and spoken language, but it's not clear if these connections are deep or superficial, profound or accidental. It is also curious that Lucky does not mention *Grammatical Man* (Simon and Schuster, 1982), Jeremy Campbell's book on information theory, language, and the coding of life.

Lucky states that part of the book's purpose is to stimulate learning about technology and its relation to our humanity and our society. The book fulfills that aim; in many ways it is stimulating, informative, thought-provoking, and downright infuriating.

First, Lucky displays complete optimism about technology and its ineluctable contribution to progress and the well-being of all people. An optimistic attitude like this is expected from engineers, who regularly tackle seemingly impossible tasks, and it no doubt makes technological advance possible. However, a book claiming to explore social aspects of information technology should better address the serious questions about progress and the effects of technology on modern life (as do Langdon Winner's *The Whale and the Reactor* (University of Chicago Press, 1986) and Samuel Florman's *The Existential Pleasures of Engineering* (St. Martin's Press, 1976)). Lucky's questions about the value of scientific and technical progress are often tucked safely away in anecdotal and fictional narratives (called "reflections") that end certain chapters.

In addition, the repeated comparisons between computer-based intelligence and human mental capacity and function affirm logic and efficiency while criticizing the redundancy and inefficiency that

characterize how people handle information-processing and communication. There is a fundamental difficulty in judging and measuring human experience in terms of the concepts and actual physical components of computing machines. Processing efficiency is not an appropriate measure of the quality of our lives, nor is it helpful for the development of what E.F. Schumacher calls "technology with a human face" (*Small is Beautiful*, Harper & Row, 1973).

Finally, the book does little to modify the common conception that engineers are out of touch with society. In discussing research on how the human brain receives and processes visual images, Lucky writes:

It is, of course, possible to inspect the wiring of a nonworking [brain] model, but there is

no real substitute for tracing the signals in one that has been fired up. Fortunately, there are monkeys who volunteer for this kind of assignment.

At face value, this statement exhibits an insensitivity that is hard to imagine in someone interested in the human effects of technology. Given the amount of discussion in the popular press and the professional literature on the proper use of animals in scientific research, it is inexcusable to say that monkeys "volunteer."

If we are to make any sense of our human experience of the development of technology, we need to get beyond facile objectification of the nonhuman world. If our modern science teaches anything, it is that objects and subjects are interrelated and that observation is intervention. We need to accept our role in experiments on

animals, as well as those we carry out daily on ourselves through our increased use of, and reliance on, computer-aided information technology. In many cases these interventions are necessary for research, but they cannot be undertaken lightly.

The book touches on many subjects — connects a few — and raises questions about nearly all of them. I wish Lucky had directly addressed the questions implicit in the anthropomorphic computer models and in the failure of humans to measure up to the technical attributes of information processing and computation. What is the human experience of information processing and technology? What is our role in its development and use?

William L. Anderson
Xerox Corporation

Computer Architecture and Design

A.J. Van De Goor (Addison-Wesley, 1989, 633 pp., \$45.25)

This book is the first of its kind I have come across and one of the few well-written books I have read recently. In spite of its title, it serves better as a book of architecture than as a book of design.

The book concentrates on microprocessors, which soon will be as powerful as today's mainframes and supercomputers. The future of hardware belongs to microprocessor technology, but there is little chance of radical philosophical changes in computer architecture. It is therefore important that any book on this topic discuss the traditional aspects of computer architecture and design.

This book focuses strongly on the Motorola 68000 architecture. The choice is generally relevant, but previous architectures might have been better suited to some places in the book. The author does describe relevant aspects of other architectures, such as the VAX, where appropriate.

The author is very knowledgeable. His systematic approach to the material divides the book into parts on the compiler interface, operating interface, acceleration, trends, and the hardware interface. These parts are then divided into chapters, sections, and subsections.

The introduction describes computer architecture and the different classifications and levels associated with it. The section on the compiler interface is well written. It discusses data representation, machine languages, addressing schemes, data operations, program flow control, and high-level program structures sup-

ported at machine level. The section on the operating interface was so good it made me want to read a entire book on operating systems. The parts on acceleration and trends introduce the reader to advanced research. The discussion of the hardware interface describes general interface schemes and associated components such as interrupts and bus arbitration.

The book is generally comprehensive, but it does not give much coverage of the computer's control unit or the basic elements on which the architecture stands. As an introductory text, the book should introduce clocking schemes, latch circuits, shifters, registers, etc. at a conceptual level so that novice readers do not have to sift through large books on digital design. Also, the exercises at the end of

each chapter are not difficult enough to truly test the reader's grasp of the material.

Although the book does not compare with other popular introductory texts, such as those by Tanenbaum, Hamacher and Zaky, and Mano, it is valuable in its emphasis on the practical aspects of computer architecture, particularly the software aspects of hardware-user interfaces. I do not recommend the book for beginners, but it is well suited for graduate study or as a reference for practicing engineers and R&D-oriented scientists. The organization, lucid presentation, and explanatory diagrams assure that computer engineers or scientists with average knowledge will benefit from the book.

Sunil Kumar Sabat
University of Southwestern Louisiana

Reviewers wanted

If you are interested in reviewing books for *Computer*, please submit your name, address, and a list of areas of interest and expertise to the Book Reviews Editor at the address below. Publishers should submit recent books for review consideration to the same address.

Guy Johnson
Department of Information Technology
Rochester Institute of Technology
1 Lomb Memorial Drive
Rochester, NY 14623

The Reliability of Expert Systems

Erik Hollnagel (Ellis Horwood Ltd., Chichester, England, 1989, 243 pp., \$44.95)

When expert systems were introduced, they were primarily seen as having potential in such areas as finance, education, and machine configuration. Initially, expert systems played noncritical roles. The decisions made or aided by these systems had very few real consequences, and their reliability was not of paramount importance.

However, as expert systems evolved, their role in decision-making and the impact of their decisions increased enormously. Expert systems are now used in medical diagnosis and nuclear power plant control. The decisions in which they are involved are no longer limited to the "blocks world" — an imaginary world, familiar to researchers in artificial intelligence, in which the only concern is the movement of blocks. Wrong information can now have disastrous repercussions. With the increased role of expert systems, their reliability has become a significant question. This book investigates reliability in depth.

The book's format is a little different from what some readers might expect. *The Reliability of Expert Systems* is the product of a 1988 seminar in Copenhagen on the safety and risks of expert systems. Each chapter represents a presentation by a different expert in the field. Some of the material does overlap, but the book is cohesive overall.

Although its focus is on expert systems, the book contains much information on software engineering, including

risk reduction, uncertainty handling, software reliability, and adequacy assurance. The history of expert systems, limitations of inference engines, complexity and failure, validations, and knowledge representation are also addressed to a lesser extent.

Readers will not need a broad background in expert systems. The book does not read like a college text, with the exception of the chapter on uncertainty. The material is presented in an easily readable manner, and references to outside sources are limited to example cases. I did find it useful to keep an AI text handy to look up terms I had forgotten over the years.

The book discusses certain contradictory ideas to avoid in expert-system design. For example, expert systems can be used in situations that the designers themselves don't fully understand, although that lack of understanding brings risk. The book emphasizes that expert systems should deal only with events that are completely understood in principle, but that are otherwise much too complex to be dealt with quickly and efficiently.

Though the book does not make any startling revelations, I recommend it to those readers who design and develop expert systems. It would be particularly useful to novices, as much of the information is common-sense knowledge that many experienced developers have already acquired.

Brian Driscoll
IBM

Implementation of Small Computer Systems: Case Studies of Applications

Richard John Whiddet (John Wiley & Sons, New York, 228 pp., \$49.95)

This book presents 13 small computer systems developed at the University of Lancaster. Fortunately, the contents do not entirely match the book's title, which suggests a strong focus on managerial issues. Each chapter describes a software or hardware project implemented by an individual or a small group. Most of the projects were executed in less than a calendar year and in academic environments, which traditionally have minimal resources.

Each chapter begins with an extensive description of the project's technical aspects. This should be welcome to readers who are not familiar with every aspect of current hardware and software technology. In this regard, the book presents not only a fresh view of the managerial aspects of implementing small computer systems but also an interesting overview of current technologies.

The book is not appropriate for traditional computer or information-science classes, but it could certainly be used in an advanced seminar for seniors or graduate students. The projects could be used as models for small projects, which students should be able to complete within either a semester or a full academic year.

The book's topics include the development of computer systems, the role of standards in computer communications and systems integration, implementation of a general-purpose database, language specification and compiler construction, and information requirements analysis.

It is apparent from the book that rigid managerial techniques are not necessary when implementing small computer systems. Moreover, it is clearly demonstrated that rigid planning can be detrimental to the success of the project. Less rigid, more creative managerial techniques can be more cost effective and provide a better final product. The book places special emphasis on using available "off the shelf" components to minimize costs and development time. Most of the chapters also mention the importance of involving users in all stages of the project, which can be done naturally when prototyping techniques are used.

Julio C. Facelli
University of Utah

Moving?

PLEASE NOTIFY
US 4 WEEKS
IN ADVANCE

Name (Please Print) _____

New Address _____

City _____

State/Country _____

Zip _____

MAIL TO:
IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08854

ATTACH
LABEL
HERE

- This notice of address change will apply to all IEEE publications to which you subscribe.
- List new address above.
- If you have a question about your subscription, place label here and clip this form to your letter.

NEW LITERATURE

Public innovation policy. In *Innovation for Failure: Government Policy and the Early British Computer Industry* (ISBN 0-262-08187-3, 264 pp., \$35), John Hendry details 10 years of effort by Britain's National Research Development Corporation to compete internationally, focusing on the creation, implementation, and management of government sponsorship policies and the response of businesses to those policies. Contact MIT Press, 55 Hayward St., Cambridge, MA 02142, phone (617) 253-2884.

Charting worldwide networks. The *Matrix: Computer Networks and Conferencing Systems Worldwide* (ISBN 0-135-65607-1, 752 pp., \$50) by John S. Quarterman provides background material and references as well as specific information on widely used networks, their interconnections, and the communities they support worldwide. Contact Digital Press, Prentice Hall, Book Distribution Center, Route 59 at Brook Hill Drive, West Nyack, NY 10995, phone (201) 767-5937.

Computer history. *Glory and Failure: The Difference Engines of Johann Müller, Charles Babbage, and Georg and Edward Scheutz* (ISBN 0-262-12146-8, 424 pp., \$45) by Michael Lindgren presents a unified picture of the difference engines that were the predecessors of the digital computer, emphasizes them as part of the history of numerical tables, and examines both their technical and social aspects. Contact MIT Press, 55 Hayward St., Cambridge, MA 02142, phone (617) 253-2884.

Software reviews and audits. Rigorous formal system evaluation is necessary before purchasing software applications or developing new in-house projects. In *Software Reviews and Audits Handbook* (ISBN 0-471-51401-2, 448 pp., \$44.95), Charles P. Hollocker, chair of the ANSI/IEEE committee on standardization of software reviews and audits, examines software requirements, preliminary review, critical design review, software verification and validation, quality system audits, in-process audits, functional audits, and physical configuration audits. The book also includes a procedures checklist for each review and audit phase as well as technical review document samples and forms. Contact John Wiley & Sons, 605 Third Ave., New York, NY 10158-0012.

Software reviews and audits. Rigorous formal system evaluation is necessary before purchasing software applications or developing new in-house projects. In *Software Reviews and Audits Handbook* (ISBN 0-471-51401-2, 448 pp., \$44.95), Charles P. Hollocker, chair of the ANSI/IEEE committee on standardization of software reviews and audits, examines software requirements, preliminary review, critical design review, software verification and validation, quality system audits, in-process audits, functional audits, and physical configuration audits. The book also includes a procedures checklist for each review and audit phase as well as technical review document samples and forms. Contact John Wiley & Sons, 605 Third Ave., New York, NY 10158-0012.

Current contents CS MAGAZINES

April 1990 IEEE Expert

Tools and Techniques — KIIS: A Prototypical Expert System for Assessing Technologies, Hamid Noori

A Skill Refinement Learning Model for Rule-Based Expert Systems, Pi-Sheng Deng, Clyde W. Holsapple, and Andrew B. Whinston

Uncertainty Management in Expert Systems, Keung-Chi Ng and Bruce Abramson

Engineering/Manufacturing — Situation Assessment for Process Control, François Jakob and Pierre Suslenschi

Government — An Expert System for Journal Selection, Roy Rada

MIS/Finance — Strategic Planning for Expert Systems, Bernard Moulin

Built-In Self-Test of the Macrolan Chip, Richard Illman and Stephan Clarke

Implementing Macro Test in Silicon Compiler Design, Frans Beekers, Rob Dekker, Rudi Stans, and Max van der Star

Serial Interfacing for Embedded-Memory Testing, Benoit Nadeau-Dostie, Allan Silburt, and Vinod K. Agarwal

A D&T Roundtable — Behavioral Description Languages, Part 2: VHDL vs. UDL/I

April 1990 IEEE Micro

Guest Editor's Introduction: The Current Japanese Computer Scene, Ken Sakamura

The Cache DRAM Architecture: A DRAM

with an On-Chip Cache Memory, Hideto Hidaka, Yoshio Matsuda, Mikio Asakura, and Kazuyasu Fujishima

Processing Element Design for a Parallel Computer, Katsuyuki Kaneko, Masatsugu Nakajima, Yasuhiro Nakamura, Junji Nishikawa, Ichiro Okabayashi, and Hiroshi Kadota

A 4-Bit, 250-MIPS Processor Using Josephson Technology, Yuji Hatano, Shimchiro Yano, Hiroyuki Mori, Hiroji Yamada, Mikio Hirano, and Ushio Kawabe

Realizing the V80 and Its System Support Functions, Hiroaki Kaneko, Nariko Suzuki, Hiroshi Wabuka, and Koji Maemura

Special Feature — Microprocessor Memory Management Units, Milan Milenkovic

April 1990 IEEE Design & Test of Computers

Guest Editorial: ITC 20th Anniversary, M.R. Mercer

Integrating Tester Pin Electronics, Christopher W. Branson

Low-Cost Testing of High-Density Logic Components, Robert W. Bassett, Barry J. Butkus, Stephen L. Dingle, Marc R. Faucher, Pamela S. Gillis, Jeannie H. Panner, John G. Petrovich, Jr., and Donald L. Wheeler

For subscription information, circle number 200 on the reader service card.

Magazine order form

Please print or use peel-off label

Single issue prices:

Member, \$10

Nonmember, \$20

Name	Quantity	Price	Total
_____	April '90	_____	_____
Address _____	Expert	_____	_____
City _____	April '90	_____	_____
State/ZIP/Country _____	D&T	_____	_____
IEEE-CS member no. (required for discount) _____	April '90	_____	_____
_____	Micro	_____	_____
Mail to: IEEE Computer Society Order Dept., 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA, 90720-1264	Total	_____	_____
	(payment enclosed)		

THE OPEN CHANNEL

"Any clod can have the facts, but having opinions is an art."
Charles McCabe, San Francisco Chronicle

Jim Haynes, Computer Center, UC Santa Cruz, CA 95065
Bitnet, haynes@ucscsc.Arpa, haynes@ucscsc.ucsc.edu

The laws of statistics

William E. Drissel, Cyberscribe Associates

(1) You never use statistics when you know what you are talking about.

Statistics are, at best, an attempt to do something a little better than throwing our hands up in the presence of uncertainty. They are useless when we have the ordinary level of certainty on which the small affairs of humanity are based. We wake up certain that Monday follows Sunday, that we are in the same house where we went to sleep. No one talks of the probability that December 25 is Christmas. The contrapositive of this law is: When a person uses statistics, he or she can't bring us ordinary certainty.

(2) No amount of computation will convert uncertainty to certainty.

The branch of mathematics dealing with probabilities arose from questions about gambling. The answers have been quite good when spread over many trials, but they offer no help about a single trial or the next flip of a coin. This law and, indeed, the value of statistics are summed up nicely by Damon Runyan: "The race is not always to the swift nor the battle to the strong, but that's the way to bet."

(3) Correlation is the weakest form of functional association, not the strongest.

Every statistics student knows the story of the man who had a hangover after drinking scotch and water one night, bourbon and water the second, and vodka and water the third. A statistician advised him to stop drinking water. This story is amusing only because our understanding of hangovers is essentially complete.

Correlation is usually used when knowledge is incomplete, and many times it leads to unsound conclusions. Thus, the correlation between malaria and swamps was correctly discerned, but the cause was incorrectly attributed to bad (mal) air (aria). The correlation between slums and high crime rates has been explained as "Slums cause crime." An alternative explanation might be that petty crime is an ineffective way to make a living — "Crime causes slums." Correlation does not distinguish between these explanations.

(4) You got a better idea?

One of the first applications of statistics to public health occurred when John Snow discovered contaminated wells in London by plotting cholera cases on a map. Because our understanding of cholera is better today, we wouldn't use such a clumsy method; we would rely on a bacteriological examination of the water. Similarly, military intelligence uses direction-finding devices to locate transmitters only until aerial photographs or observer sightings are available.

(5) Any treatment improves the worst.

Galton found that the children of extremely tall parents tended to be taller than average but not as tall as their parents. He called this "regression to the mean." The word "regression" has stuck with studies of this type ever since. Apparently, the height of extremely tall people is due to a systematic, inheritable component plus a random component that is not inherited.

Similarly, a member of the population who exhibits "worst" behavior over time might exhibit systematic, enduring badness plus random badness that happened to be very bad during the sample period. Applying a treatment to the worst — even if the treatment has no effect on the systematic badness — might seem to show an improvement because the random badness during the next sample period might not be as severe.

Consider a concrete example: Pick the most drought-stricken area of your country for 1989. Chances are this area is dry every year because of its location. It's the worst in 1989 because of that enduring effect plus the random weather effects during 1989. To help the farmers of that region, mumble an incantation. If you check in the fall of 1990, you will probably find that you eased the drought.

(6) If it ain't statistically significant, it ain't nothin'.

Companies often use statistics to claim "Our Product is better than Brand X." When statisticians study questions like

this, they commonly test whether the samples of Our Product and Brand X overlap so much that we can't tell them apart. If so, the statisticians say there is no statistically significant difference. The average wear of Our Tires might seem better, but because the lives of individual tires overlap with those of Brand X, the statistician concludes that the difference in the averages is due to chance.

This means comparative data must be accompanied by some indication of scatter such as standard deviation. Sample size is also helpful. A common statistical method uses two averages, two standard deviations, and two sample sizes to decide the significance of the difference between the two sample averages. I severely discount comparative conclusions unaccompanied by sample sizes and some indication of scatter. Bear this in mind the next time you read that boys (girls) "have been shown" to be better at some task. Reproductive activities aside, the scatter in abilities among men (women) far overwhelms the differences between men and women.

(7) No matter how you slice 'em, they're still statistics.

Cell studies of how broken bones knit are medical knowledge. Statistical data about new or uncertain treatments are statistics. Chemical changes in teeth are dental knowledge; cavity rates are statistics. No matter how suggestive the data are for practitioners, as long as we are uncertain of the underlying science, engineering, or prevalence of a phenomenon, our data are statistics. And statistics — regardless of the subject matter — are still statistics.

The doctor, engineer, physicist, or politician does not come equipped with knowledge or background in sampling statistics. Yet, for some reason, doctors present medical statistics to the public, politicians present political statistics, and so on. Conclusions based on statistical data and presented by subject-matter experts should be viewed skeptically if there is no indication the statistics have been vetted.

Why is there no Statistician General?

IEEE INTERNATIONAL CONFERENCE on WAFER SCALE INTEGRATION

Fairmont Hotel
San Francisco, California U.S.A.
January 29-31, 1991

Sponsored by the **IEEE Computer Society**
and the **IEEE Components, Hybrids, and Manufacturing Technology Society**

General Chair: Michael J. Little Hughes Research Laboratories
Program Chair: Professor Vijay K. Jain University of South Florida

The IEEE International Conference on Wafer Scale Integration is the premier forum for reporting new WSI devices and progress in WSI design, Fabrication, and packaging technologies. Papers directly related to WSI are sought from workers around the world. In selecting papers for the Conference, preference will be given to papers that describe working WSI devices and systems. Exhibition of operating hardware at the conference is strongly encouraged. Submissions are invited in the following areas:

- System Applications of WSI
- WSI Implementations
- WSI Architecture
- WSI Devices
- WSI Packaging (Incl. Thermal Design)
- I/O Approaches (Electrical/Optical)
- Clock, Signal, and Power Distribution
- Testing and Testability
- Defect Tolerance/Yield Enhancement
- Fault Tolerance/Self Repair
- CAD for WSI
- Silicon on Silicon Hybrids

Six copies of paper summaries including the name and mailing address of the preferred contact are due by June 15, 1990. The summary should be sent to:

Submissions from the Americas

Professor Vijay Jain
Dept. of Electrical Engineering
University of South Florida
Tampa, FL 33620
Phone: (813) 974-4741

Submissions from Europe

Professor R. Mike Lea
Brunel University
Uxbridge UB8 3PH
UNITED KINGDOM

Submissions from Asia

Dr. Tadashi Sasaki
Sharp Corporation
8, Ichigaya, Hachiman-cho
Shinjuku-ku, Tokyo 162
JAPAN

To be placed on the Conference mailing list, please send your name and address to:

In the Americas and Asia:

Terry Chappell
WSI Conference Registrar
730 Encino Drive
Aptos, CA 95003
Phone (408) 662-1936

In Europe:

Professor R. Mike Lea
Brunel University
Uxbridge UB8 3PH
UNITED KINGDOM
Phone (+44) 895-74000 ext. 2821
FAX (+44) 895-58728
E-Mail: Mike.Lea@brunel.ac.uk



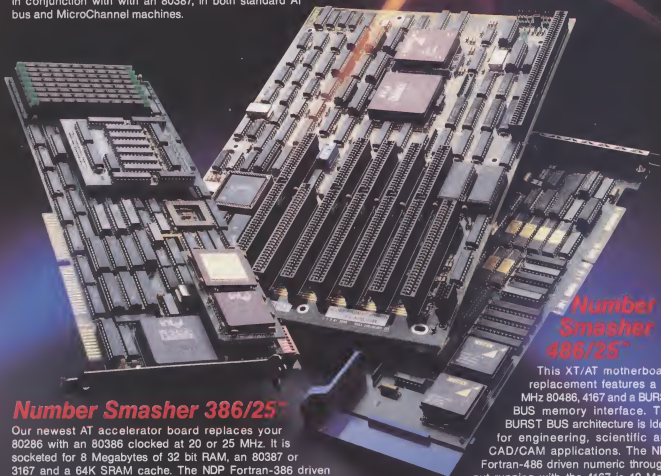
IEEE COMPUTER SOCIETY



THE INSTITUTE OF ELECTRICAL AND
ELECTRONICS ENGINEERS, INC.

Weitek Your 386/486!

The new 4167 delivers up to 10 Megaflops when driven by NDP Fortran-486 and is supported by dozens of scientific, engineering and CAD applications. MicroWay provided the tools to develop many of these applications and supplies the interface cards required to use Weitek coprocessors in conjunction with with an 80387, in both standard AT bus and MicroChannel machines.



Number Smasher 486/25*

This XT/AT motherboard replacement features a 25 Mhz 80486, 4167 and a BURST BUS memory interface. The BURST BUS architecture is ideal for engineering, scientific and CAD/CAM applications. The NDP Fortran-486 driven numeric throughput running with the 4167 is 12 Megawhetstones and 10 Megawhetscales (see BYTE 1989 IBM issue).

Number Smasher 386/25*

Our newest AT accelerator board replaces your 80286 with an 80386 clocked at 20 or 25 Mhz. It is socketed for 8 Megabytes of 32 bit RAM, an 80387 or 3167 and a 64K SRAM cache. The NDP Fortran-386 driven 3167 throughput at 25 Mhz is 5.5 Megawhetstones.

mW3167/387

This popular daughterboard (shown on the Number Smasher 386/25) lets you plug a 3167 and an 80387 into a 386 system that has a single EMC socket.

3167/4167 Numeric Performance

	3167/MCA	NS 386/25	NS/486/25
Megawhetstones	3.4	5.5	12.2
Megawhetscales	1.6	3.1	9.9

mW3167/MCA

Our MCA Weitek card runs in the IBM Model 70 and 80. At 20 Mhz, its performance is 2 to 3 times that of an 80387.

NDP Fortran-486 and C-486 are globally optimized main-frame compilers that have been fine tuned for the 80486 and 4176.



Weitek 3167 and 4167 are trademarks of Weitek Corp., MicroWay and Number Smasher are registered trademarks of MicroWay, Inc., 80386, 80387, 80486 are trademarks of Intel Corp.

MicroWay

World Leader in PC Numerics

Corporate Headquarters: P.O. Box 79, Kingston, MA 02364 USA (508) 746-7341
32 High St., Kingston-Upon-Thames, UK, 01-541-5466

USA FAX 508-746-4678 Italy 02-74.90.749 Holland 40 836455 Germany 069-75-2023

Reader Service Number 5